Procesamiento y muestra de los KPIs de rendimiento con gráficas mediante una Base de Datos MySQL y una Interfaz web

Artem Motsarenko

Ingeniera de Telecomunicación (Planificación y gestión de las Telecomunicaciones)

Tutor: Jesús Arias Fisteus

Director: Myriam Escobar Zapardiel

Leganes (Madrid)

Julio de 2015

Universidad Carlos III de Madrid

Dedico a mi familia por apoyarme durante todos estos años y a mi abuela que no llegó a ver esto pero la que siempre confiaba en mi.

Agradezco a Jesus Arias Fisteus por realizar la labor de ser tutor de este proyecto.

Agradezco a todos mis compañeros de la universidad por aguantarme estos años y compartir conmigo los momentos más difíciles y más felices. A Ulises, David, Manu, Adri, Silvia, Jesus, Ivan, Jeff, Javier, Daniel. Especialmente a mi compañero y amigo Ulises.

Agradezco a mis compañeros de Vodafone de ambos equipos en los que he estado trabajando durante estos meses y a mi manager Enrique Sanz.
Especialmente quiero agradecer a las chicas de "Red Planet" (Alicia, Carol, Patri) por los momentos divertidos que hemos pasado durante estos meses y por vuestros ánimos.

Finalmente agradezco a mi tutora en Vodafone y directora de este proyecto Myriam Escobar. Gracias por darme esta posibilidad de realizar el proyecto y por todo aprendido durante estos meses tanto lo relacionado con el trabajo como con la vida en general.

Resumen

Trabajando en entornos de grandes empresas es muy importante tener tanto la visibilidad global de los procesos que están sucediendo como poner el foco en detalles más pequeños que muy a menudo pueden tener un impacto muy importante en el trabajo que estamos realizando si se les presta suficiente atención. Por ello, es siempre importante disponer de herramientas que permitan monitorizar más detalladamente diferentes procesos para entender mejor qué está sucediendo, qué impacto puede tener y qué medidas se deben tomar para mejorar el rendimiento del equipo, departamento, área o empresa en general. Aparte de esto, en grandes empresas donde la información que puede gestionar un equipo viene de diferentes fuentes y con diferentes formatos, es muy importante centralizar diferentes herramientas de monitorización para hacer el proceso de análisis de datos más cómodo y más rápido.

Quizás sobre todo por estas razones, surge la necesidad de realizar el proyecto que se describe en esta memoria. Básicamente, el proyecto resuelve el problema de manejo de grandes volúmenes de información y el problema de visualizar el comportamiento que representa esta información de una manera cómoda y facil. Los datos que se manejan en este proyecto recogen toda la información sobre los cambios de tarifa que se realizan en los centros de atención de llamadas.

Para resolver el problema de manejo de grandes volúmenes de datos se implementa una Base de Datos MySQL que se irá actualizando mensualmente con nuevos datos que llegan desde otro departamento. Estos datos en cierta medida ya vienen preprocesados y simplificados, aunque requieren un procesado y un agrupamiento adicional por parte de nuestro equipo antes de cargarlos a la BBDD.

Por otro lado tenemos una interfaz web que resuelve el problema de visualización de esta información almacenada en nuestra BBDD mediante generacion y muestra de diferentes graficas. Para llevar a cabo esta tarea los dos bloques, la BBDD y la interfaz web, se interconectan entre sí. La interfaz web invia hacia la BBDD consultas ques especifican los datos requeridos en cada momento, y la BBDD devuelve estos datos que posteriormente se transforman en diferentes gráficas.

Contenido

1.- Introducción

1.1. - Motivacion

A finales de octubre de 2014 me incorporo como becario a un pequeño equipo recién creado dentro del Departamento de Operaciones Comerciales (COPS) de Vodafone España.
La función principal del departamento es proporcionar el servicio de atención al cliente del sector particular de los clientes de Vodafone España. Esta tarea en sí requiere una multitud de tareas de seguimiento, diseño, tecnología, medición y previsión que realizan diferentes equipos dentro del departamento.

Algunas de la tareas de los equipos de previsión es estimar el posible volumen de llamadas de clientes a los centros de llamada de Vodafone en cada momento, haciendo estimaciones de volumen de llamadas con detalle de por mes, semana, día e incluso hora. Este tipo de previsiones es muy complejo ya que hay que hacer estimaciones en función del segmento del cliente, acontecimientos eventuales como podría ser un partido de fútbol importante o lanzamiento de una campaña comercial, repartiendo posibles volúmenes en diferentes centros de llamada que gestionan el servicio. Una mala parametrización de este servicio tiene un impacto muy importante para el departamento, ya que puede conllevar al aumento de tiempo de espera por parte del cliente o, por ejemplo, a que un cliente sea atendido en la plataforma que no le corresponde, en cuyo caso el agente no le podrá atender del mismo modo que si fuese un agente que realmente corresponde a este tipo de llamada. Este tipo de cosas pueden influir negativamente en la percepción del servicio por parte del cliente.

Por otro lado, los equipos de medición se dedican a medir una multitud de KPIs (Key Performance Indicators) relacionados con el servicio de atención al cliente. Empezando por los KPIs que reflejan la percepción del servicio por parte de cliente y acabando por los KPIs de parametrización, haciendo el desglose por días, semanas, centros, etc. Algunos de estos KPIs son, por ejemplo, volúmenes de llamadas que se atienden antes o después de X segundos, volúmenes de llamadas que caigan, volúmenes de llamadas que por congestión en la red acaban en una plataforma o un servicio que no les corresponde, etc.

Equipos de tecnología se dedican al diseño y continua mejora de los procesos de enrutamiento de las llamadas de cliente. Su función es conseguir clasificar el motivo de llamada dentro de los posibles motivos (información general, incidencia, factura, terminal, etc) y conseguir que la llamada acabe atendida en el menor tiempo posible por el agente que mejor podrá gestionarla.

Los equipos restantes del departamento tienen un carácter más comercial, ya que sus tareas están relacionadas con todo que tiene que ver con la comunicación entre cliente y agente. Aparte de objetivo principal que es atención al cliente y resolución del motivo de llamada, existe una componente comercial cuyo objetivo es, una vez analizado el perfil del cliente, ofrecerle productos y servicios que le puedan interesar. Estos equipos gestionan información y productos que se le ofrecen al cliente en las llamadas, diseñan modelos de atención, diseñan productos y servicios, gestionan diferentes campañas comerciales que se realizan en las llamadas, coordinan la comunicación con los centros de llamada, etc.

Dentro de estos equipos se sitúa el equipo al que yo me uní. El equipo tiene varias funciones principales:

- Gestión y seguimiento del rendimiento (de los KPIs) de las agencias y coordinación de su trabajo con los responsables de las mismas.
- Gestión, diseño, parametrización y rendimiento de las NBAs (Next Best Actions) de los clientes.
- Diseño y gestión del recomendador de tarifas que utilizan los agentes en sus llamadas.
- Seguimiento de los ingresos que supone toda la actividad comercial de los centros de llamada del segmento particular.

Debido a que nuestro equipo es un equipo recién formado, todas estas tareas en mayor o en menor medida son tareas nuevas para los integrantes del equipo, lo que supone una continua mejora y diseño de nuevos procesos y mejoras de los mismos. Adicionalmente, debido a un entorno muy dinámico y nuevas necesidades de seguimiento, surgen tareas y procesos nuevos.

El trabajo diario de mi equipo, y el mío en particular consiste entre otras cosas en obtención de los datos su actualización y procesado para poder llevar correcto seguimiento del rendimiento de los distintos KPIs, ver posibles influencias debidas a acciones tomadas, campañas comerciales, posibles incidencias, etc., para tener una visión global.

Llevar a cabo estas tareas de una forma eficiente supone mucha interconexión y coordinación con otros equipos y departamentos. Además, para poder recibir, procesar y distribuir la información de una forma eficiente creamos y utilizamos muchas herramientas que permitan automatizar estas tareas (scripts, macros, cuadros de mando, etc).

Es de aquí de donde finalmente surge la necesidad de diseñar la herramienta que estoy presentando como el proyecto fin de carrera. Básicamente, la herramienta tiene que permitir el manejo de ficheros de gran de tamaño, cuyo manejo es imposible de realizar con las hojas de cálculo como Excel, y teniendo en cuenta que cada mes se van acumulando nuevos datos, llegará un momento en el que incluso una herramienta tan potente como Access nos pondrá un límite. De aquí viene la necesidad de utilizar Bases de Datos (BBDD) de MySQL utilizando MySQL Server.

Otra de las necesidades es intentar unificar la visualización del mayor número de los KPIs posibles en un único sitio para facilitar el acceso a los mismos por parte de las personas implicadas. De aquí surge la idea de crear un sitio web que nos permita ubicar en él los resultados de diferentes consultas a las BBDD anteriores y representar estos resultados en forma de diferentes gráficos.

Estos KPIs que vamos a visualizar, principalmente representan la información relativa a los cambios de tarifas de los clientes que se realizan en los centros de atención de llamadas. Algunos de los más importantes KPIs que se trata de representar son por ejemplo los volúmenes de cambios que se realizan a partir de una tarifa origen, hacia una tarifa destino o pareja de mismos. También otro punto muy importante, es que estos KPIs los podemos visualizar por centro o por servicio especifico de cada uno de los centros y de manera evolutiva, lo que nos permite visualizar el comportamiento de cada uno de estos y compararlo con los comportamientos de los demás.

1.2.- Objetivos

Plantear objetivos concretos, continuando con la línea de la motivación. Puede ser una sección muy breve con simplemente una lista de objetivos.

Los objetivos del proyecto se pueden dividir en dos bloques. Por un lado tenemos los objetivos para el cliente y por otro lado objetivos técnicos necesarios para conseguir los objetivos del cliente.

Los objetivos del cliente son muy simples, y ya he hablado por encima sobre ellos:

- Posibilidad de visualizar diferentes KPIs que provienen de volúmenes muy grandes de datos.
- Rapidez de procesamiento y de muestra de los resultados.
- Unificación de la visualización de los KPIs en un sitio único.
- Conseguir que el uso de la herramienta sea lo más fácil posible para el cliente, es decir sin necesidad de que tenga algunos conocimientos específicos relacionados con BBDD o interfaces web.

Objetivos técnicos para conseguir los objetivos planteados arriba:

- Diseñar y montar una BBDD MySQL con MySQL Server teniendo en cuenta la estructura de los ficheros origen con los que vamos a trabajar.
- Plantear el proceso de carga y de actualización periódica de los datos.
- Diseño de un sitio web (la parte que va ver el usuario).
- Interconexión del sitio web con nuestra BBDD.
- Diseño de las consultas en MySQL para extraer los datos necesarios en cada momento. Implementación de estas consultas mediante código PHP.
- Representación de los resultados en forma de gráficos.
- Debido al entorno dinámico que puede suponer cambio del formato de datos origen o la necesidad de visualizar datos nuevos, nuestro diseño debe permitir realizar estas modificaciones sin provocar grandes cambios o problemas para el programador.

Esquema de la herramienta a alto nivel:

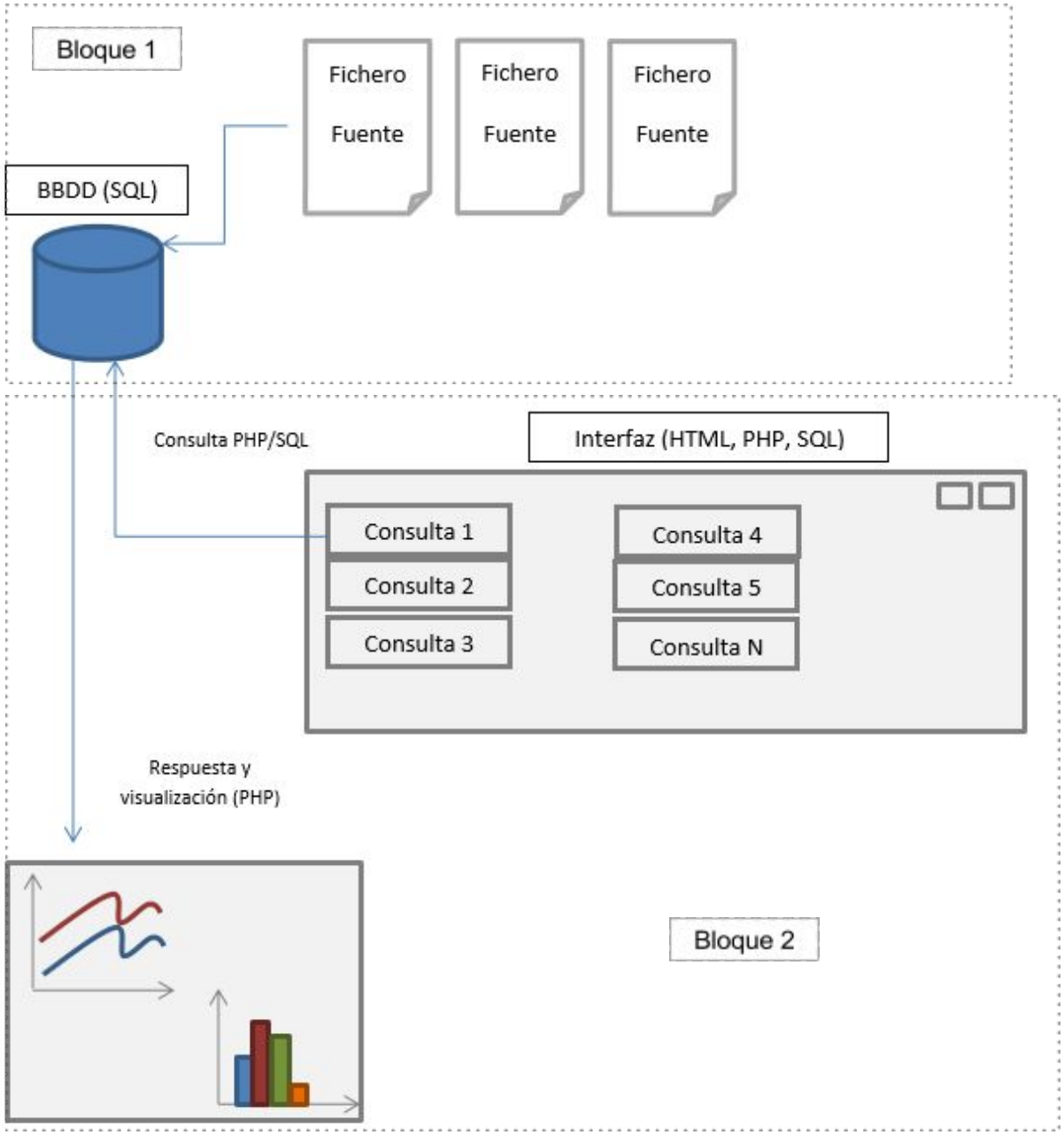

Ilustración 1: Esquema de la herramienta a alto nivel

1.3.- Plan de trabajo

Plantear un muy breve resumen del plan de trabajo y referirse al capítulo del plan de trabajo.
Diagrama de Gantt (p.e. como los de Microsoft Project)
Recuerda incluir estudio del problema y documentación + redacción de la memoria

El plan de trabajo está separado en siete fases, algunas de estas fases van en paralelo con otras. Voy a describir brevemente cada una de ellas:

- Fase0:
  Esta es la primera fase en la que surge la necesidad de conseguir una solución a las necesidades que surgen. En esta fase primer paso que hacemos es definir claramente que problemas queremos resolver, para de este modo poder definir una arquitectura a un alto nivel y empezar a analizar posibles alternativas. Una vez analizadas las alternativas elegimos las que mejor nos convengan. De este modo ya tenemos definida una arquitectura de alto nivel sobre la que empezamos a trabajar.

- Fase1:
  Una vez que tenemos definida nuestra arquitectura a alto nivel y hemos elegido la posible solución para nuestros objetivos empezamos a profundizar en más detalle en cada uno de los bloques que forman parte de nuestra arquitectura. En esta fase en concreto empezamos con el Bloque1 que está relacionado con nuestra BBDD que vamos a utilizar.
  Para empezar el desarrollo de esta primera fase instalamos todos los softwares necesarios y realizamos la configuración de los mismos. Antes de empezar cualquier diseño o programación, tenemos que asegurarnos que las versiones de distintos softwares y que hemos instalado son compatibles entre sí. En el caso contrario en algún momento del desarrollo del proyecto nos puede surgir el problema de la incompatibilidad de las versiones lo que supondrá la reinstalación de las mismas y posibles pérdidas de configuraciones y diseños realizados.
  Una vez que tengamos nuestras herramientas instaladas y configuradas empezamos con el diseño y la creación de nuestra BBDD. Siguiente paso dentro de esta fase será definición, diseño y prueba de posibles métodos para carga y actualización de los datos de nuestra BBDD así como para la extracción de los mismos.

- Fase2:
  Fase de desarrollo de la Interfaz web (Bloque2). Para la realización de esta fase necesitamos instalar los softwares necesarios según el diseño que hemos definido. Antes de empezar con el diseño de la web necesitamos definir los objetivos básicos que queremos que cumpla nuestra interfaz. Una vez que los tengamos procedemos a realizar un diseño inicial de la página que a medida del desarrollo del proyecto irá sufriendo cambios e irá creciendo. Siguiente paso muy importante dentro de esta fase es configuración de la conexión entre nuestra interfaz web y nuestra BBDD. Esta conexión será precisa para poder realizar cualquier consulta a nuestra BBDD desde la interfaz web o realizar cualquier modificación en la BBDD.

- Fase3:
  Una vez que en la Fase2 hemos conseguido establecer la conexión entre la interfaz web y nuestra BBDD, podemos empezar esta fase que consiste en desarrollar esta interconexión mediante integración de código PHP y HTML con el código SQL.
  Según los datos que vamos a querer tener en cada momento necesitamos desarrollar nuestras consultas en SQL para lanzar a nuestra BBDD. Empezaremos con consultas simples, y a medida del desarrollo del proyecto iremos afinando estas consultas para que los datos extraídos sean lo más precisos posibles. La precisión de los datos nos permitirá reducir el procesado de datos en el lado de la interfaz web, con esto ganaremos el tiempo de ejecución lo que permitirá que los resultados se irán mostrando más rápido.
  El objetivo principal de esta fase es obtención de los datos desde la BBDD integrando las consultas de SQL dentro del código PHP.

- Fase4:
  Una vez tengamos los primeros datos extraídos desde nuestra BBDD podemos empezar a pensar cómo vamos a mostrar estos datos en forma de los gráficos de diferentes tipos y que necesitamos para esto. Para poder pintar gráficos existen muchas librerías ya creadas que nos permitirán ahorrar el tiempo. En esta fase analizaremos posibles librerías de PHP que podemos utilizar y elegiremos una de ellas para el desarrollo.
  Esta fase solapa con la Fase3 ya que depende de los datos obtenidos con nuestras consultas que hemos diseñado. Adicionalmente se solapa con la Fase2, ya que a medida de que vayamos obteniendo primeras graficas podemos empezar a pensar como y donde las queremos colocar. Esto supondrá cambios y reorganización de la estructura de la interfaz web ajustándola a las necesidades.

- Fase5:
  Es la última fase técnica del proyecto. Consiste en la realización de simulaciones de todo el ciclo de utilización de la herramienta, desde la carga o actualización de los datos hasta que podamos ir cambiando de páginas dentro de la nuestra interfaz para visualizar distintos resultados.
  Se supone que en este punto los objetivos del proyecto están alcanzados y posteriores modificaciones de los bloques del proyecto se realizarán para las posibles mejoras sugeridas o para introducir cambios sugeridos por parte de los usuarios.

- Fase6:
  La última fase del proyecto consiste en la redacción del documento de la memoria del proyecto.

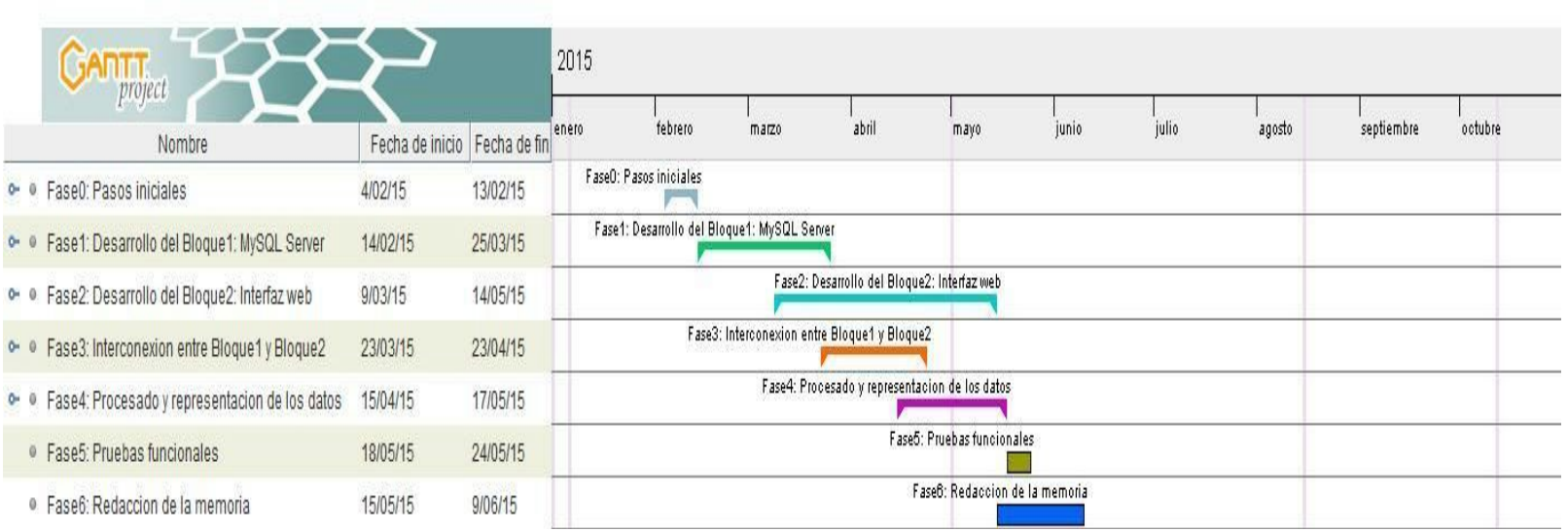

Ilustración 2: Diagrama simplificada de la planificación:

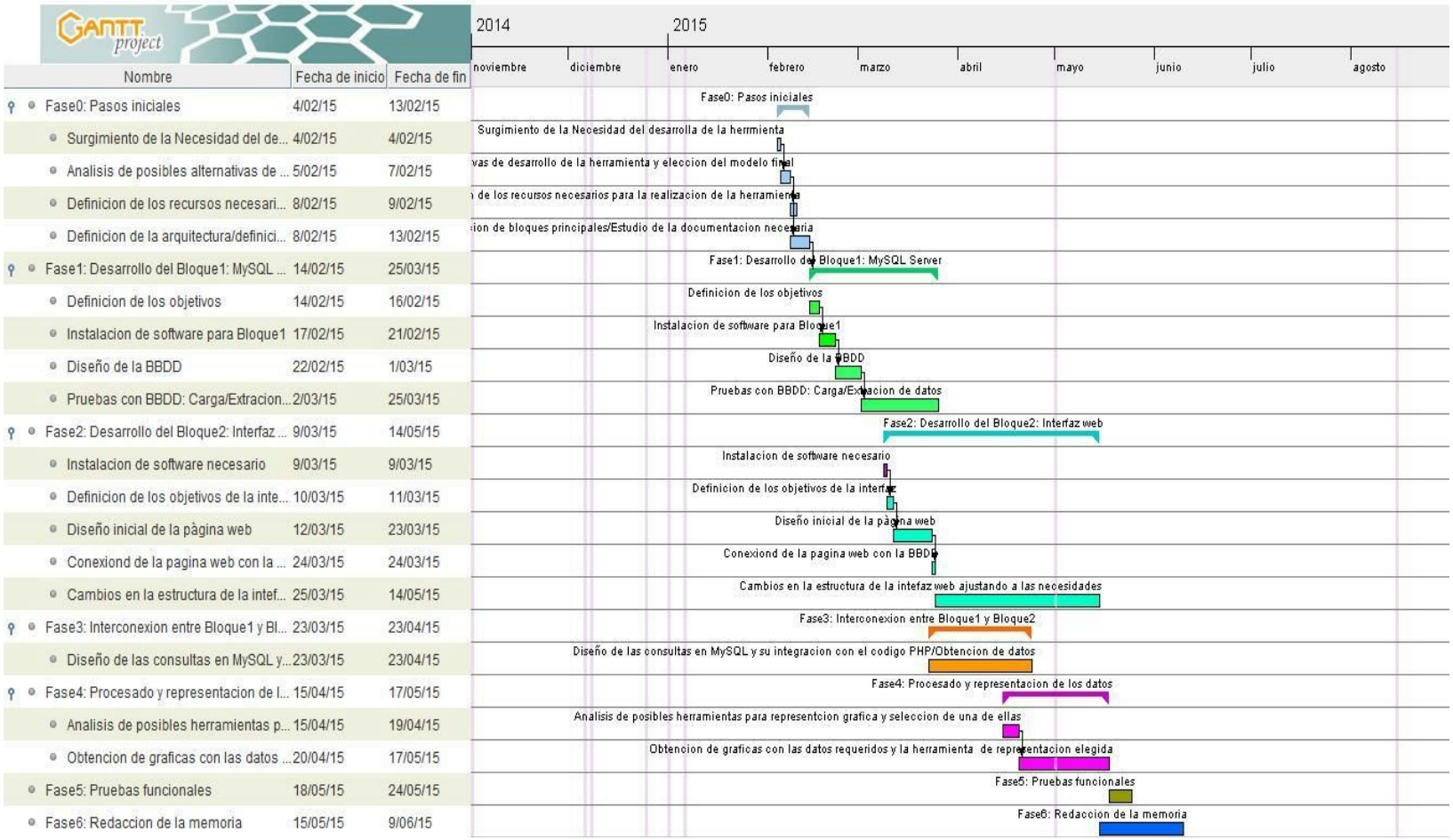

Ilustración 3: Diagrama detallada de la planificación

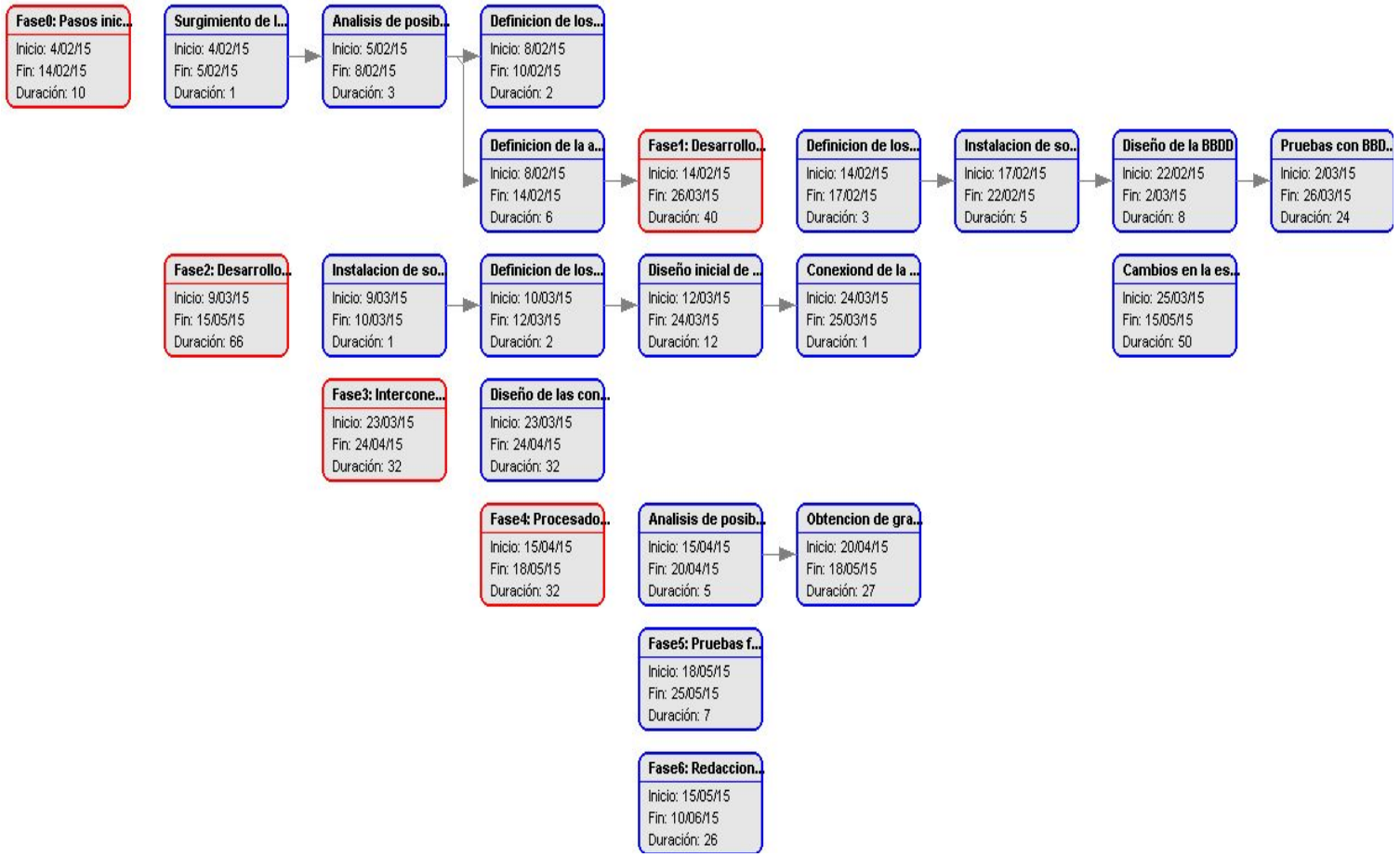

Ilustración 4: Diagrama de Planificación en bloques

1.4.- Introducción al resto de la memoria

En este punto quiero hacer una pequeña introducción al resto de la memoria de mi proyecto comentando un poco de que irá cada uno de los capítulos haciendo una breve descripción de cada uno de ellos.
En el capítulo dos, Estado del Arte, haré una introducción al mundo de CRM (*Customer Raltionship Mangemenet*) comentando en que consiste la tecnología, que objetivos y qué importancia tiene para las empresas en el mundo actual. Intentaré en la medida de lo posible colocar mi proyecto dentro de la explicación de CRM para demostrar utilidad real de la misma. Además hablaré de algunas de las más famosas aplicaciones de CRM que existen en actualidad y comentare las herramientas CRM que utilizamos en nuestro trabajo comentando las funcionalidades que tienen y para que se utilizan.

Después nombrare muy brevemente las funcionalidades que queremos que tenga nuestra herramienta, es decir definiremos los requisitos del sistema.

En el capítulo de cuatro, Arquitectura del sistema, presentare la arquitectura principal de nuestro sistema, hablaré sobre los bloques que tienen y cómo están unidos entre sí. También hablaré sobre los principales procesos que se ejecutan dentro del sistema permitiendo que la misma funciona. Para entender todo esto mejor proporcionaré un esquema del sistema mostrando los principales procesos.

Como continuación al capítulo anterior, en el capítulo cinco hablaré sobre el diseño de la herramienta. En este capítulo entraré más en detalle de cada uno de los bloques explicando su diseño. Hablaré como están hechos estos bloques y los métodos más importantes que contienen.

Una vez explicado el diseño y funcionalidad que tienen los bloques, en el capítulo de la implementación hablaré sobre los aspectos más relevantes del diseño y sobre las dificultades de implementación que he tenido a la hora de realizarlos. Pondré algunos ejemplos de los mismos explicando cómo se han resuelto finalmente.

En el capítulo siete comentaré las pruebas que se han hecho para comprobar la funcionalidad correcta del sistema. Aquí hablaré de las pruebas para ver que realmente se cumplen todos los requisitos y también haré pruebas para medir los tiempos de respuesta del sistema en diferentes puntos.

Una vez que la implementación del nuestro sistema está finalizada y el sistema funciona, podemos sacar conclusiones del proyecto viendo el cumplimiento de los requisitos que hemos puesto al principio del proyecto. Además viendo como ha quedado el sistema y teniendo primeros feedbacks de los usuarios podemos plantear posibles trabajos futuros y posibles mejoras para aumentar la funcionalidad del mismo. De todo esto hablaré en el capítulo ocho.

Cualquier proyecto tiene que tener un presupuesto, sobre el presupuesto de nuestro proyecto se trata en el capítulo nuevo. Comentaré todo lo necesario para la realización del proyecto, es

decir mano de obra, softwares instalados, etc., comentando el precio de cada una de las partes y haciendo un precio final.

Finalizará con una lista de acrónimos utilizados a lo largo de la memoria, tabla de ilustraciones que aparecen a lo largo de la memoria y adjuntando una lista de referencias bibliográficas utilizadas para citar a diferentes fuentes y autores.

2.- Estado del arte (antecedentes)

2.1.- CRM

CRM (Customer Relationship management) son las siglas que los últimos años podemos escuchar muy a menudo, sobre todo si trabajamos en entornos empresariales o en organizaciones. En seguida hablaré sobre el CRM y sobre sistemas y herramientas de CRM que utilizamos nosotros en nuestro equipo para nuestro trabajo diario.
El propio nombre CRM, traduciéndose al castellano como el Manejo de Relaciones con el Cliente, ya por sí mismo nos dice mucho sobre esta estrategia. Ante de todo tenemos que entender que CRM no es una herramienta informática, por lo que para la empresa que quiere implantar CRM en su trabajo no basta con solo instalar herramientas de CRM. CRM engloba mucho más significado, es una forma y una filosofía de trabajo que se centra en el cliente y en sus necesidades, y todo esto se consigue utilizando nuevas tecnologías y nuevas sistemas de la información.

CRM es:

"Una estrategia centrada en el cliente, que busca un crecimiento en beneficios a través de proporcionar un mayor valor al cliente." (Rieger 2010).

Esta definición Rieger explica el significado, pero quizás es un poco corta y no refleja todo lo que supone CRM, por lo que me gustaría citar también siguiente definición:

"Proceso de construcción y conservación de relaciones rentables con los clientes, mediante la entrega de un valor superior y de una mayor satisfacción. Las empresas modernas van más allá del diseño de estrategias para atraer a nuevos clientes y realizar transacciones con ellos. Estas emplean la gestión de la relaciones rentables y duraderas con ellos." (Kotler 2011)

La importante de esta definición es que Kotler subraya la importancia de estrategia de conservación de relaciones con clientes ya existentes y no solo el desarrollo de estrategias de captación de los nuevos. Creo que es uno de los principales cambios en las estrategias de las empresas, por lo que empresas ponen mucho foco actualmente en la gestión de los clientes ya existentes para satisfacer todas sus necesidades y mantener su actual cartera de clientes. Nuestro trabajo que realizamos en Vodafone es un ejemplo claro de esta estrategia.
Muchos de los autores en sus definiciones, asocian CRM a la tecnología, a las aplicaciones y a bases de datos que son capaces de manejar y procesar grandes volúmenes de la información.

“Básicamente consiste en centrar tu modelo de negocio en el cliente y dotar a tu empresa de las herramientas técnicas que permitan presta un servicio y comunicación a tus usuarios, inmejorable. Este medio nos ofrece una oportunidad única para utilizar la tecnología en nuestro favor, y generar un conocimiento del cliente que difícilmente se alcanza en otros canales. Esto además de permitirnos crear mucho más valor, nos ayudará a crear una importante ventaja competitiva. Se trata de establecer una relación a largo plazo con nuestros clientes. Es una estrategia de negocio orientada a la fidelización de clientes.” (Master-Net, Dicionario técnico 2010)

La tecnología tiene un papel muy importante en el éxito de CRM. Muchos de los procesos de CRM serían imposibles de realizar sin nuevas herramientas y tecnologías que han surgido en los últimos años, y quizás por ello la estrategia no surgió antes. Procesamiento y gestión de grandes volúmenes de datos serían imposibles sin las BBDD que podemos utilizar hoy en día. A lo largo de este proyecto se hablará sobre la utilización de las BBDD y las ventajas que proporcionan.

Resumiendo un poco todo lo dicho anteriormente, podemos definir finalmente CRM como: “Una estrategia de aplicación que facilita organización y procesamiento de las ventas, procesos de marketing más personalizados al cliente, procesos de atención al cliente y procesos de almacenamiento y de procesado de la información con la finalidad de dar un mejor servicio al cliente, beneficiando de este modo tanto al cliente como a la compañía.”

Una de los lemas de las soluciones CRM es la importancia que tiene el cliente y la personalización que se le da el cliente. No todos los clientes son iguales, por lo que cada uno de ellos tiene necesidades específicas, requiere una atención y una oferta específica. Para poder realizar esta gestión específica se requiere un análisis continuo de clientes y de mercado que no solo permita entender las necesidades de clientes sino también permita predecirlas. Todas estas tareas se realizan básicamente por el departamento de Marketing, que debe gestionar toda la información relativa a esto.
Es evidente que en compañías que gestionan millones de clientes, como la compañía telefónica en mi caso, a veces no es fácil o incluso es imposible personalizar miles de necesidades de todos los clientes. De aquí surge la necesidad de segmentación de los clientes y de segmentación de la atención al cliente.
Los clientes se segmentan en función de sus características dentro de la compañía, como por ejemplo empresas, clientes particulares, prepago y pospago o incluso por tipo de productos que tienen contratados. Por otra parte, los servicios de atención a clientes también se segmentan en función de las necesidades de los clientes, habiendo de este modo servicios de atención que gestionan diferentes necesidades como resolución de las necesidades técnicas, resolución de incidencias, facturación, contratación de productos, etc. Todo esto permite una atención mucho más personalizada de cliente, ya que por segmento del mismo ya se puede predecir de cierto modo posibles motivos de llamadas y dirigirle al servicio con los agentes especializados.

El siguiente punto muy importante que me gustaría mencionar son los canales de atención y de comunicación. En el mundo en el que vivimos actualmente este punto es muy importante ya que continuamente surgen nuevas tecnologías y nuevos medios de comunicación como redes sociales a los que personas, y por supuesto clientes, dedican cada vez más tiempo que a los medios de comunicación clásicos.

Hablando sobre los canales de atención al cliente, por la específica del sector de la empresa en la que he realizado el proyecto, se destacan sobre todo los centros de atención telefónica. Aparte de ellos podemos destacar también la atención al cliente en los canales presenciales, es decir, en tiendas, y atención por medio de Internet. En este caso se trata sobre todo sobre autogestión de los clientes. En el rendimiento de los centros de atención telefónica y en parte en los de centros de atención presencial se basa el presente proyecto.

Volviendo al tema de los medios de comunicación con el cliente, cabe destacar la importancia que tienen actualmente las redes sociales y la facilidad que dan ellas para comunicarse con el cliente. Esto influye en que las soluciones CRM actuales evolucionan continuamente incorporando cada vez más la información de los clientes disponible en las redes sociales.

Normalmente los softwares de CRM que utilizan grandes e importantes empresas cuestan dinero, aunque existen opciones de softwares CRM llamadas de Código Abierto. Estos softwares son menos costosos o incluso se les puede adquirir de forma gratuita. Además como estos softwares son de código abierto se les puede modificar ajustándose a las necesidades de cada empresa. En general son una buena solución para pequeñas empresas o para empresas que quieren probar por la primera vez un software CRM y obtener primera experiencia. Existen proveedores de software CRM de código abierto para diferentes tamaños y modelos empresariales. (Chris (buscocrm) s.f.)

Algunos más famosos de ellos son los siguientes:

- SugarCRM: líder del mercado de los softwares CRM de código abierto creado en el año 2004. Se ha convertido en un estándar para los softwares CRM de código abierto. El software permite la administración de la relación con los clientes. La aplicación está basada en el código PHP que interactúa con una BD del Servidor MySQL. Software muy popular en el mercado de pequeñas empresas que permite crear relaciones estables con los clientes y aumentar las ventas. Existen varias versiones de dicho softwares con distintas funcionalidades según el precio, sin embargo también existe una versión gratuita. (CRMespañol s.f.) (sugarCRM s.f.)
- vTiger: herramienta muy usada en los mercados de desarrollo. Incorpora la automatización de la Fuerza de Ventas, servicio al cliente, gestión de provisionamiento, automatización de marketing, formularios de la web, etc. Gestiona los procesos de venta y calcula las probabilidades de éxito según el historial lo que permite ayudar a los comerciales saber que estrategia seguir con cada cliente. (Chris (buscocrm) s.f.)
- SplendidCRM: una solución que se construye sobre la plataforma de Microsoft con un Servidor SQL. Este software es dirigido a los canales indirectos de consultores, revendedores de valor añadido y a los integradores del sistema. La principal diferencia de este software de los demás, es que no está desarrollada sobre el Linux sino sobre plataforma Windows con acceso al código de la fuente. (Chris (buscocrm) s.f.)

Proyectando un poco la estrategia de CRM a nuestro trabajo, quiero recordar que el trabajo que realizamos dentro de nuestro departamento y en nuestro equipo en particular, es la monitorización del rendimiento de los centros de atención de llamadas telefónicas. Tradicionalmente en muchas empresas se consideraba este servicio como exclusivo para la atención de clientes y para la resolución de las necesidades de cliente. De este modo este servicio suponía un coste para la empresa. Sin embargo, una buena y correcta implementación de la estrategia CRM puede conseguir no solo recuperar este coste sino también obtener beneficios de este servicio, ya que un incremento en la satisfacción de los clientes con el servicio que reciben por parte de la compañía debe repercutir en un incremento de ventas que se realizan en este canal de atención.

Hablando sobre las herramientas CRM, me gustaría mencionar algunas de muchas de ellas que utilizamos a diario. Estas herramientas nos permiten manejar la información, realizar diferentes procesos relacionados con gestión de clientes, realizar comprobaciones de diferentes tipos, comprobar estado de procesos o tareas, etc.
Estas herramientas en el primer lugar se dividen por el segmento de cliente que permiten gestionar, es decir, particulares, empresas, miniempresas o Pymes.
Siendo nuestro departamento responsable de la gestión de los clientes particulares, las herramientas que utilizamos son específicas para la gestión de este tipo de clientes. A su vez, las herramientas de gestión de clientes particulares se dividen básicamente en dos tipos, las que gestionan a clientes de pospago y las que gestionan a clientes de prepago. Aparte de ellas existen también herramientas mixtas, que se utilizan para la gestión de toda la cartera de clientes. Algunas de estas herramientas son por ejemplo las que realizan consultas de estado de cobertura en diferentes zonas o localidades, las que manejan toda la información relacionada con cualquier tipo de incidencia, y las herramientas que permiten obtener y monitorizar la información más técnica y más detallada de las llamadas que se atienden, como por ejemplo la información sobre el enrutamiento, tiempo de respuesta, etc.
A continuación hablaré más en concreto sobre cuatro de estas herramientas que utilizamos más a menudo, comentando para que sirvan y que tipo de gestiones hacemos con ellas.

Empezaré por la descripción de las herramientas de gestión de clientes, "*Herramienta de gestión clientes prepago*" y "*Herramienta de gestión clientes pospago*". Ambas son producto de la marca Siebel Systems, que es una compañía de gran relevancia en el mercado de diseño, desarrollo, comercialización y soporte de software de CRM. Ambas herramientas tienen la misma funcionalidad y el mismo objetivo con la diferencia de que una de ellas gestiona a la cartera de clientes prepago y la otra a los de pospago. Esta diferencia en el tipo de cliente supone ciertas diferencias en los procesos de gestión y atención de los mismos que también se queda reflejada en las herramientas. Básicamente las diferencias son debidas a los temas de permanencia, oferta y provisionamiento, por lo que los clientes de pospago requieren más gestiones de este tipo. Ambas herramientas son las herramientas principales para utilizar por los agentes de atención telefónica. Contienen todo la información relativa a los clientes, empezando por datos personales y terminando por los productos contratados por el cliente, especificaciones de los mismos o ciclo de facturación que tiene el cliente. Cuando el agente recibe una llamada de cliente, la herramienta carga de modo automático la ficha del cliente que está llamando según el número del cliente, permitiendo al agente resolver muchas de las posibles necesidades del cliente. Además, estas herramientas permiten la gestión de contratación de diferentes productos y las provisiones necesarias. Por ejemplo, según la necesidad de cliente el agente puede cambiar la tarifa que tiene el cliente en alguna de sus líneas asociadas y provisionar el envío de un terminal a la dirección del cliente si el mismo solicita renovar el terminal.

Como he dicho antes, son las herramientas principales para el uso por parte de los agentes. Siendo nuestra función principal la monitorización del trabajo y del rendimiento de los centros, y de los agentes en particular, utilizamos estas herramientas cuando queremos ponernos en el lugar de los agentes para entender cómo trabajan y realizan las tareas. Además de esto, las utilizamos cuando queremos consultar los datos de algún cliente específico o cuando queremos comprobar cómo y qué datos se cargan cuando se atiende a un cliente específico. Podríamos decir que esta herramienta realiza la función de la ventana principal que ve el agente, por lo que cada agente atiende solo a su perfil de cliente, pospago o prepago. Cuando el agente necesita realizar ciertas consultas o acciones, esta herramienta carga los datos o se interconecta con más herramientas. Una de estas herramientas a partir de la cual carga los datos la herramienta descrita es la herramienta que proporciona toda la información sobre la oferta relativa al cliente, la describiré en el siguiente punto.

Siguiente herramienta, “Herramienta de asignación de ofertas”, es la herramienta que maneje la información de ofertas que se pueden ofrecer a un cliente. Estas ofertas pueden ser de tipo general, personal para cada uno de los clientes o para un segmento o para un grupo de clientes según las características de los mismos. Es debido a que cada segmento de clientes tiene sus necesidades y hay que saber muy bien qué ofertas y cuándo ofrecer a cada uno de ellos. Esta evaluación de las necesidades y la generación de las ofertas las realizan básicamente departamentos de *Marketing* y de *Customer Value Management* (*CVM*) en función del segmento del cliente, consumo que realiza el cliente, productos que tiene contratados, campañas puntuales, etc. La información de la herramienta se actualiza periódicamente con nuevas ofertas y nuevas recomendaciones. Esta información aparece en un panel adicional, y en función de ella los agentes pueden ofrecer o recomendar al cliente cierto producto o servicio. Para que el agente pueda visualizar esta información, la herramienta principal descrita en el punto previo se conecta con la herramienta de ofertas, y esta a su vez en función del segmento del cliente y de su número de teléfono le devuelve diferentes ofertas.

Última herramienta que me gustaría mencionar es el “*SAP Business Objects*”. Quizás es una herramienta más de Business Intelligence que directamente de CRM, pero es una herramienta clave que utilizamos para diferentes procesos de CRM que realizamos a diario para monitorizar ciertos KPIs y detectar tendencias en el rendimiento de centros y de los agentes.
Es una herramienta a través de la web, por lo que permite ser utilizada por elevado número de usuarios y de forma segura, ya que requiere autenticación y permisos para ser utilizada. La herramienta permite explorar grandes volúmenes de datos facilitando el acceso a las BBDD. Gracias a que permite almacenar la información de una manera organizada y estructurada, podemos realizar de forma intuitiva diferentes consultas de datos y generar múltiples tipos de informes y paneles de control.
La información que obtenemos desde esta herramienta es relativa a las llamadas que entran en los centros y que se atienden por los agentes. Por lo que la utilizamos básicamente para monitorizar el número de llamadas que se atienden y monitorizar otros KPIs de rendimiento comercial tanto globales por centros como detallando por grupos de trabajo o por agente.

2.2.- BBDD

Esta sección se trata sobre las Bases de Datos, sus definiciones, tipos y alternativos que existen en la actualidad. Con el propio nombre ya podemos entender el significado de una BD. Es un lugar donde podemos almacenar nuestros datos, pero no es tan simple. Existen muchas definiciones de las BBDD que dan una explicación más exacta y más completa de qué es realmente una BBDD:

"*Fondo común de información almacenada en una computadora para que cualquier persona o programa autorizado pueda acceder a ella, independientemente de su procedencia y del uso que haga*" (Olga Pons, y otros 2005)

Creo que esta primera definición define por así decir las necesidades que queremos resolver con una BBDD, por lo que necesitamos completar un poco más el concepto de una BBDD:

"*Colección o depósito de datos, donde los datos están lógicamente relacionados entre sí, tienen una definición y descripción comunes y están estructurados de una forma particular. Una base de datos es también un modelo del mundo real y, como tal, debe poder servir para toda una gama de usos y aplicaciones*" (Conference des Statisticiens Europeens 2012)

La primera definición hace más referencia al tema de acceso a la información almacenada en la BBDD, y la segunda la concreta con el modelo de esquema lógico que organiza los datos almacenados. Para terminar con la explicación del concepto de BBDD me gustaría poner la siguiente explicación de la misma por parte de IBM, que además la relaciona con un ejemplo simple:

"*De una manera simple, es un contenedor que permite almacenar la información de forma ordenada con diferentes propósitos y usos. Por ejemplo, en una base de datos se puede almacenar información de diferentes departamentos (Ventas, Recursos Humanos, Inventarios, entre otros). El almacenamiento de la información por sí sola no tiene un valor, pero si combinamos o relacionamos la información con diferentes departamentos nos puede dar valor. Por ejemplo, combinar la información de las ventas del mes de junio del 2014 para el producto 'X' en la zona norte nos da un indicativo del comportamiento de las ventas en un periodo de tiempo.*" (Morales 2014)

Ahora bien, una vez tenemos claro qué es en sí una BBDD, surge la pregunta de ¿cómo se gestionan esas BBDD por parte de usuario? La gestionan se realizan a través de Sistemas de gestión de Base de Datos o simplificando SGBD (DBMS en inglés). SGBD consiste en un conjunto de programas que permiten acceder a la BBDD y manipular los datos que almacena. Podemos decir que SGBD funciona como una interfaz entre la BBDD y el usuario, permitiendo a este acceder a los datos, almacenarlos, modificarlos, realizar unas extracciones de los mismos, construir y definir nuevas tablas de datos dentro de una BBDD. A parte de esto los SGBD tienen mecanismos para mantener la integridad de los datos, es decir control de concurrencia, que permite a varios usuarios acceder a la BBDD y realizar cambios de una forma controlada. (Foster y Godbole 2014)

El proceso de interacción entre el usuario y la BBDD lo podemos resumir en el siguiente esquema:

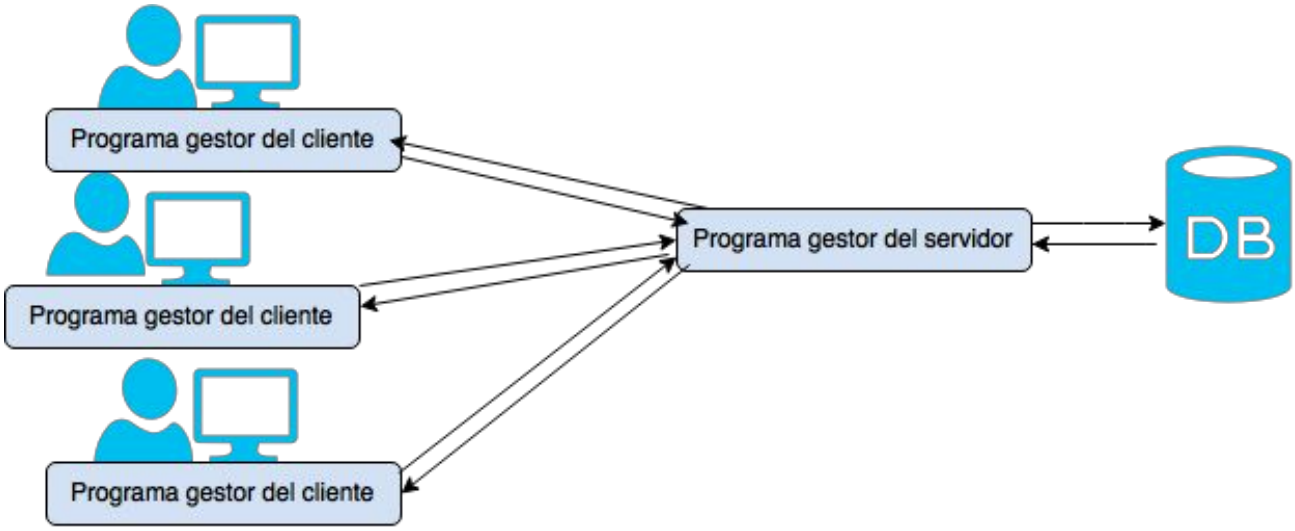

Ilustración 5: Esquema de interacción Cliente-BBDD

En actualidad existen muchos tipos de BBDD, según el contenido de datos que almacenen o según el uso que se les va a dar. Básicamente podemos definir cinco tipos de BBDD:

- **Bases de Datos Jerárquicas**. Este tipo de BBDD almacena la información de una forma jerárquica similar a un árbol. Los elementos dentro del árbol se llaman nodos y pueden tener o no tener *hijos*. Los nodos que tienen *hijos* se llaman nodos *padre*, los que no tienen *hijos* se llaman nodos *hojas* y finalmente el nodo que no tiene *padres* se llama nodo *raíz*. Este tipo de BBDD permite crear estructuras estables para el manejo de grandes volúmenes de información. (J. 2013) (Stephens 2008)

- **Bases de Datos en Red**. Este modelo es parecido al modelo anterior, pero la gran diferencia es que permite que un nodo *hijo* tenga varios *padres*. Modelo en Red resuelve el problema de redundancia de datos que tiene el modelo Jerárquico, pero aun así el modelo no es usado muy a menudo por los usuarios normales, ya que supone ciertas dificultades en la administración de la información. (J. 2013) (Stephens 2008)
- **Bases de Datos orientadas a Objetos**. Es uno de los modelos más recientes de la BBDD. Se almacena tanto el estado de un objeto como su comportamiento en forma de operaciones. Es decir, los usuarios pueden definir operaciones que se pueden realizarse sobre los datos como parte de la definición de la BBDD. Este tipo de BBDD se diseña para trabajar con lenguajes de programación orientados a objetos como es el caso de Java, Visual Basic, etc. (J. 2013) (Stephens 2008)
- **Bases de Datos Multidimensionales**. Como el propio nombre dice son las bases de datos con varias dimensiones, es decir, los datos en vez de almacenarse en tablas bidimensionales se almacenan en tablas multidimensionales. Son muy útiles para proyectos donde tenemos que almacenar grandes volúmenes de información y también para aplicaciones de procesamiento analítico. (J. 2013) (Stephens 2008)
- **Bases de Datos Relacionales**. Es una estructura de Bases de Datos que permite realizar gestiones de forma dinámica. Los objetos están relacionados entre sí dentro de las tablas que contienen filas (registros) y columnas (campos). La información en este tipo de tablas se almacena y se consulta mediante unas consultas simples y muy flexibles. El lenguaje más utilizado para este tipo de bases de datos es el lenguaje SQL. Podemos decir que es el modelo de Bases de Datos más utilizado hoy en día. (J. 2013) (Stephens 2008)

El modelo elegido para la realización de este proyecto es el de Bases de Datos Relacionales. Este modelo como he dicho ya antes es el más usado y presenta muchas ventajas como:

- Ofrecen sistemas simples para la manipulación y representación de los datos.
- Integra herramientas que permiten evitar la duplicidad de los datos.
- Garantiza la integridad referencial entre los registros
- Uniformidad en la manipulación de los datos por parte de los usuarios.
- Sencillez de comprender y utilizar por parte del usuario.

Existen muchos BBDD que siguen el modelo relacional, los más famosos y más utilizados son MySQL, Oracle, Derby, PostgreSQL, MariaDB, etc. Para este proyecto hemos elegido la opción de MySQL En el capítulo “4.-Arquitectura del sistema” hablaremos más sobre esta opción y sobre las ventajas que proporciona. (DuBois 2013)

2.3.- Desarrollo de aplicaciones web

En esta sección hablaré sobre las aplicaciones web, las tecnologías actuales para el desarrollo de las mismas y concretaré con la tecnología PHP, que es la que fue utilizada en este proyecto.

Para empezar me gustaría recordar la definición de una aplicación web:

"Una aplicación web es un programa informático que en lugar de ejecutarse en un ordenador personal (una aplicación de escritorio), se ejecuta parcialmente en un servidor remoto, al que se accede a través de Internet por medio de un navegador web". (Moreira Gibaja 2009)

Creo que esta definición describe completamente el significado de una aplicación web. Es decir, es una aplicación que se ejecuta en el ordenador del servidor, lo que supone que los datos sobre los que trabajamos están almacenados y procesados en el servidor web y no en nuestro ordenador local. Ya que decimos que estas aplicaciones se ejecutan en el ordenador del servidor significa que estas aplicaciones no necesitan ser instaladas en nuestro ordenador para poder utilizarlas, simplemente tenemos que acceder al sitio web donde están ubicadas.

Una de las ventaja de estos consiste en que si queremos modificar o actualizar esta aplicación cargada en el servidor, simplemente tenemos que modificarla allí sin necesidad de distribuir e instalar nuevas versiones en los ordenadores de todos los usuarios que la utilizan. Para los usuarios esto supone muchas comodidades, ya que no necesitan preocuparse por licencias y actualización, la aplicación se actualiza en un único sitio y para todos usuarios. Además los usuarios no tienen que preocuparse por posible incompatibilidad de versiones de los sistemas operativos, ya que todo se procesa y se ejecuta en la máquina del servidor y el usuario solo visualiza los resultados de este trabajo. De este modo estas aplicaciones no consumen los recursos hardware de nuestro ordenador como lo hacen las aplicaciones de escritorio clásicas.

Por otro lado, son aplicaciones multi-usuario, lo que significa que pueden ejecutarse por múltiples usuarios que acceden a ellas. Los usuarios solo necesitan tener una conexión a internet y en algunos casos permisos o datos de acceso como usuario y contraseña para acceder y para ejecutar las aplicaciones. (Lemay y Colburn 2010)

Resumiendo un poco todo dicho hasta el momento, presento el siguiente esquema que muestra a alto nivel como funciona una aplicación web del tipo que hemos utilizado en este proyecto.

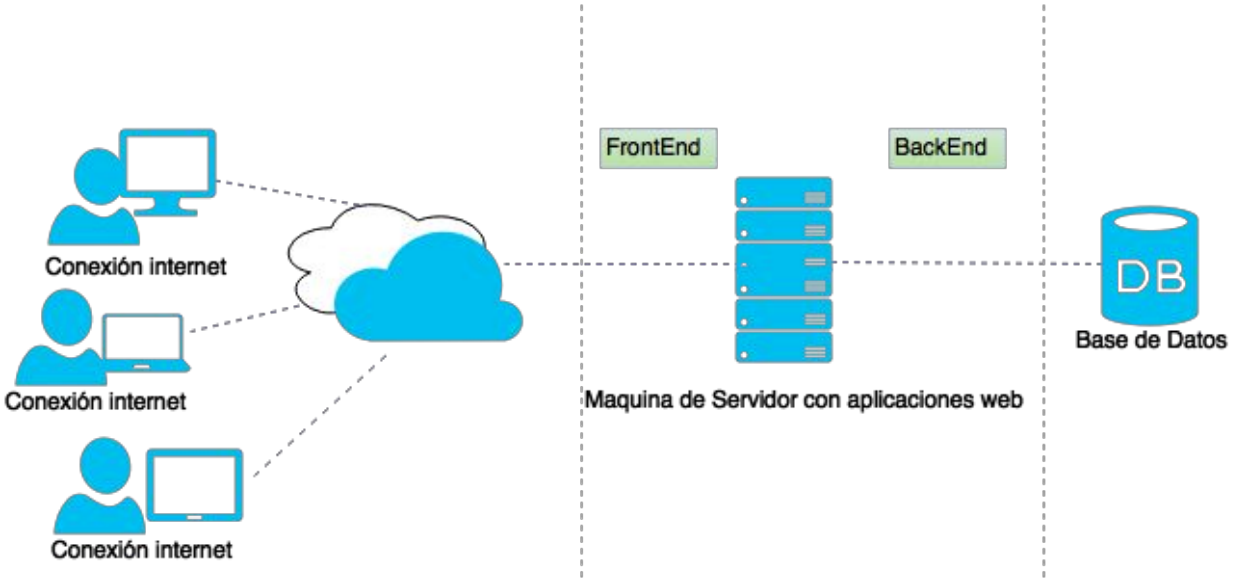

Ilustración 6: Aplicación web

Como vemos la primera capa es básicamente el navegador web que nos permite la conexión a internet. La segunda fase está constituida por una máquina que ejerce de servidor, capaz de utilizar una tecnología dinámica como Java Servlets, ASP o PHP como en el caso de nuestro proyecto. Y finalmente como la tercera capa tenemos una BBDD donde se almacena la información necesaria que utiliza la aplicación web. Es un ejemplo claro de la estructura dinámica en el lado del servidor. A continuación explico con más en detalle el aspecto del dinamismo en las páginas.

El desarrollo de las aplicaciones está evolucionando continuamente en los últimos años con la creación de nuevas tecnologías, librerías para el desarrollo, frameworks para el desarrollo más fácil, etc. Básicamente podemos dividir las páginas y aplicaciones web entre dinámicas y estáticas, tanto en el cliente como en el servidor. Antes de explicar que significan cada una de ellas, me gustaría explicar los significados de páginas web estáticas y dinámicas:

- Una página web estática muestra contenidos permanentes, es decir, son las páginas de tipo más informativo que interactivo, ya que no permiten ningún tipo de interacción por parte del usuario más allá de los enlaces a otras páginas u hojas. Para diseño de este tipo de páginas se utiliza el lenguaje HTML, de aquí entendemos HTML como un lenguaje de programación web estática. La ventaja de estas páginas es la facilidad de desarrollo que tienen.
- Una página web dinámica permite introducir elementos de interacción con el usuario que visita la página. Estas iteraciones se llaman eventos y se generan a partir de las acciones realizadas por el usuario en la página. Como ejemplo de mismos podemos pensar en los botones que puede pulsar el usuario para realizar alguna acción. En la realización del proyecto se utilizan estos tipos de botones, por lo que los veremos en más adelante. La creación de un sitio web dinámico es más compleja, ya que aparte de lenguaje HTML para el diseño de bloques estáticos dentro de la página dinámica, requiere conocimientos de otros lenguajes de programación que introducen elementos dinámicos. Dentro de estos lenguajes los más usados son PHP y JavaScript. Las ventajas de estas páginas son evidentes, ya que permiten introducir en la página mucha más funcionalidad y hacerla más atractiva para el usuario que la visita.

Ahora bien, una página dinámica en el lado del cliente permite darle al usuario más opciones de funcionalidad, pero también significa que toda la carga de procesamiento de estos elementos de funcionalidad cae sobre el navegador del cliente. Estos efectos dinámicos en el lado del cliente se crean con el código JavaScript que se introduce dentro del código HTML utilizado para el diseño de la página en la que están. Cuando el navegador abre una página que contiene elementos dinámicos, se encarga de ejecutarlos para proporcionar estas funcionalidades al cliente.

Por otro lado la página dinámica en el lado del servidor es interpretada y ejecutada en el propio servidor. La principal ventaja es la información centralizada y más protegida a la que pueden acceder una multitud de usuarios. Se implementa para el desarrollo de estas páginas, aparte de HTML, código PHP que da la funcionalidad dinámica. Cuando un cliente accede a este tipo de páginas, el servidor ejecuta todo el código generando la página final con los resultados que ve el usuario. En este caso el servidor maneja toda la información que se almacena en las BBDD.

Cabe destacar que una página puede ser dinámica en el cliente, en el servidor o dinámica en ambos lados. En este proyecto la aplicación web desarrollado presenta sobre todo características dinámicas en el lado del servidor, ya que todo el procesamiento y la generación de los resultados se realiza allí, pero también tiene elementos dinámicos en el lado del cliente en forma de botones.

Como he dicho antes, para introducir los elementos dinámicos en el lado del cliente se utiliza básicamente JavaScript. El lenguaje se ejecuta en las máquinas virtuales en los navegadores, lo que permite una velocidad alta de ejecución. Además existen muchas librerías de JavaScript que se pueden utilizar en el desarrollo de aplicaciones. Unas tecnologías parecidas, aunque menos usadas, son Flash de Adobe y ActiveX de Microsoft. Todo esto convierte JavaScript en la mejor alternativa para la introducción de los elementos dinámicos en el lado del cliente.

Ahora bien, para el desarrollo en el lado del servidor sí que tenemos muchas alternativas a elegir. Existen muchas tecnologías para el desarrollo de aplicaciones en el lado del servidor y cada vez aparecen nuevas. Las más famosas son PHP, ASP, ASP.NET, JSP, Python, Ruby, django, etc. Cada una de estas tecnologías presenta sus ventajas y sus inconvenientes y cada una de ellas es más apropiada para diferentes tipos de proyectos.

Como he dicho ya en las secciones anteriores, la tecnología y el lenguaje que se utiliza en este proyecto es el PHP. Es el lenguaje de programación del lado del servidor con el que se ha implementado el mayor número de servidores en el mundo. Esto gracias a las muchas ventajas que tiene tales como:

- Lenguaje de código abierto.
- Lenguaje muy rápido y muy fácil de aprender (sintaxis parecida a C y Java).
- Es un lenguaje multiplataforma (soporta Windows, Linux, etc.).
- Lenguaje orientado a desarrollo de aplicaciones web dinámicas.
- Permite la conexión con la mayoría de las BBDD.
- Dispone de una gran librería de funciones y ejemplos.
- Multitud de frameworks disponibles (Zend, CakePHP, etc).
- Mucha documentación disponible.

Todo esto convierte PHP en una de las mejores opciones para el desarrollo del sitio web dinámico. Unos ejemplos de páginas web famosos que utilizan esta tecnología como soporte de sus aplicaciones son Facebook, Yahoo, Wikipedia, etc. Además PHP es utilizado por famosos CMS (Content Management System) como WordPress y Drupal. A parte de esto la pareja de PHP y MySQL es muy utilizada para el desarrollo de sitios web. (Lorna 2013) (Wenz 2012)

3.- Requisitos

Enumeración de requisitos (sin necesidad de explicarlos en detalle). Puede ser una enumeración.

- Requisito 1:
  Implementar mecanismos de la visualización del rendimiento de los principales KPIs del departamento, tales como recomendador, NBA (*Next Best Action*), ingresos con los datos consultados desde la BBDD diseñada.

- Requisito 2:
  Unificar la visualización de los KPIs en un sitio único lo que permitirá al usuario un análisis más cómodo evitando consultas a otras herramientas y cuadros de mando

- Requisito 3:
  Permitir la interconexión e la interacción entre los principales bloques del sistema que permita el funcionamiento efectivo del sistema.

- Requisito 4:
  Permitir y facilitar el acceso a la herramienta por parte del personal implicado, implementando una interfaz web.

- Requisito 5:
  Diseñar las consultas SQL y el mecanismo del envío de las mismas hacia la BBDD desde nuestra interfaz web

- Requisito 6:
  Facilitar el uso de la herramienta por parte del usuario, haciéndola intuitiva y permitiendo su interconexión con las demás herramientas del departamento.

- Requisito 7:
  El sistema debe permitir manejo de grandes volúmenes de datos que se actualizan mensualmente teniendo en cuenta la perspectiva a largo plazo.

- Requisito 8:
  Diseño que permita introducir cambios acordes de las necesidades cambiantes del entorno dinámico de trabajo.

- Requisito 9:
  Incluir mecanismos que automaticen o simplifiquen el proceso de carga y de actualización de los datos.

4.- Arquitectura del sistema

Grandes bloques del sistema y cómo se comunican/relacionan entre sí (figura más explicación textual).
Puede ser a varios niveles de detalle y/o en varias fases.
Justifica todo lo que puedas, diseños alternativos desechados,...

En esta fase explicaré más profundamente la arquitectura del sistema, aunque sin entrar en muchos detalles técnicos sobre el diseño, ya que los aspectos más relevantes del mismo se presentarán en el siguiente capítulo. Hablaré brevemente sobre los principales procesos dentro de cada bloque, sobre cómo están unidos los bloques y cómo interaccionan entre sí. Con este capítulo pretendo explicar cómo funciona el sistema desde el paso inicial que es la carga de datos hasta llegar al objetivo del proyecto, visualizar diferentes gráficas.
Además de esto iré explicando en la medida de lo posible las herramientas utilizadas y por qué las he elegido.
Para empezar a explicar la arquitectura definiré los dos bloques principales de la herramienta que forman parte del sistema:

- Bloque 1: Es el bloque de la BBDD MySQL. Este bloque incluye todo lo relacionado con la creación de la base de datos, su diseño, su actualización y su configuración. Podemos decir que este bloque forma parte del *backend*. Es decir, el usuario no lo ve y no sabe cómo funciona y quizás ni que exista.
- Bloque 2: Es el bloque de la interfaz web que incluye todo el diseño de la interfaz web para ajustarlo a las necesidades del proyecto. Es realmente la parte con la que interactúa el usuario, es el *front-end* de la herramienta.
- Lo demás, los procesos de interconexión, creación y envío de consultas, recepción y procesamiento de los datos y la representación de las gráficas, forman parte también del diseño. Son procesos de interacción entre los dos bloques principales, por eso no están dentro de uno u otro. Para mayor facilidad podemos pensar que forman parte de un Bloque de Procesamiento.

Una vez hecha esta pequeña introducción presento la arquitectura del sistema e iré explicando diferentes procesos numerados dentro de la arquitectura explicando para que se utilicen y a que bloque pertenecen.

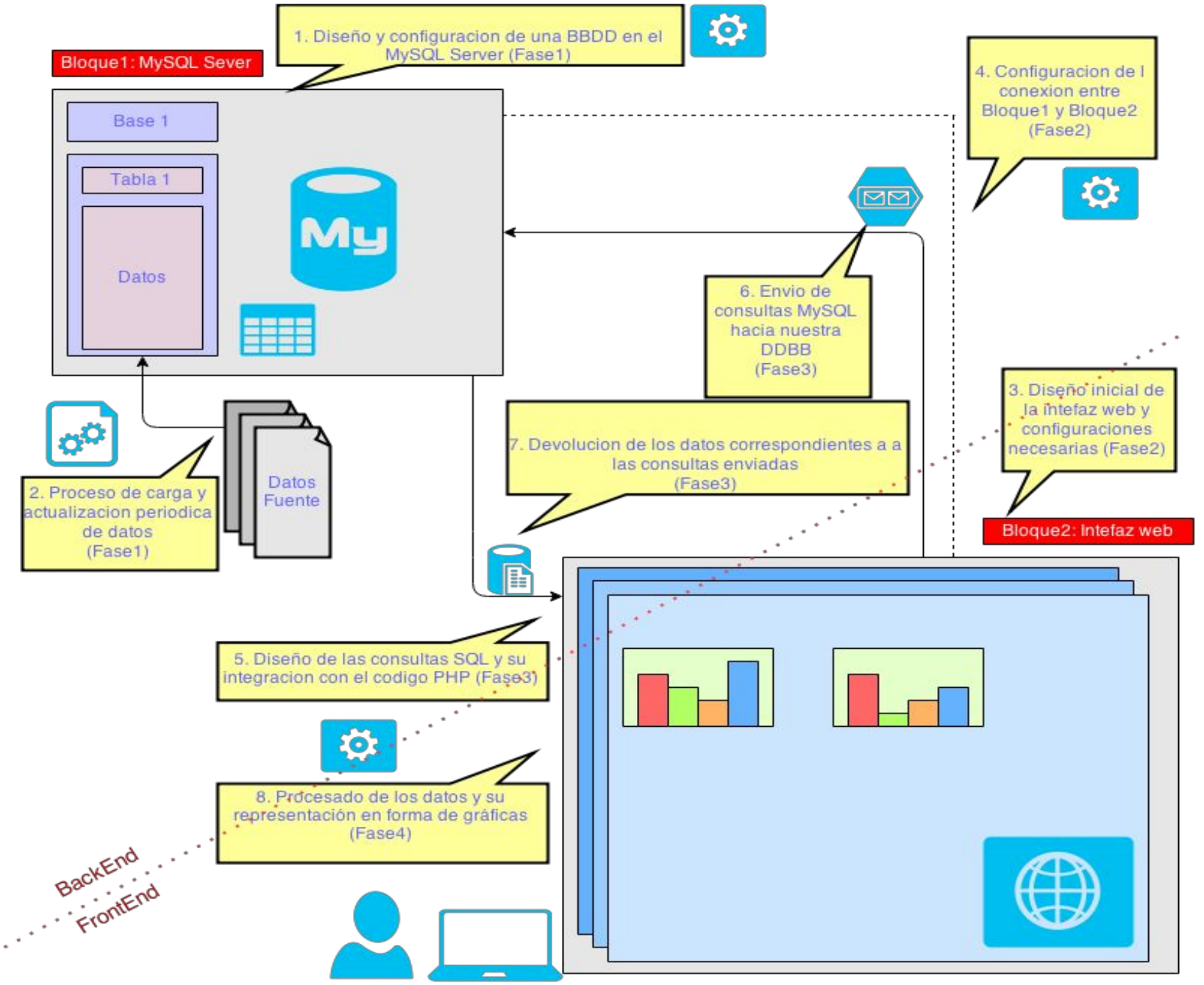

Ilustración 5: Arquitectura del sistema

Empezando por la explicación del Bloque 1 la primera cosa que deberíamos hacer es explicar por qué hemos elegido MySQL como BBDD para nuestro proyecto. Principalmente la razón fue que para nuestro proyecto necesitábamos una BBDD de código abierto. Existen muchas posibilidades de utilizar BBDD de código abierto, pero finalmente he elegido MySQL por simplicidad que tiene. Por otra parte MySQL tiene numerosas ventajas como por ejemplo:

- Acceso de forma simultánea a la BBDD por parte de los usuarios. Para nosotros esto significará básicamente que si varios usuarios están utilizando la herramienta al mismo tiempo no tendrán problemas al solicitar diferentes gráficas que por debajo requieren consultas a la BBDD. Es una de las mejores opciones para el desarrollo de proyectos web.
- Permite gestionar de una forma fácil permisos y privilegios de diferentes usuarios según el rol que van a desempeñar.
- Fácil instalación y configuración de la BBDD.
- La velocidad al realizar las operaciones es bastante rápida y se ajusta a las necesidades del nuestro proyecto.

Se puede hablar mucho más sobre las ventajas de MySQL, pero las ventajas que realmente nos importan para la realización del proyecto son las citadas arriba. Otro punto que hace la utilización de MySQL más atractiva y más fácil es la existencia de herramientas adicionales para la gestión y la administración de la base de datos como MySQL Administrator y otra para la creación y ejecución de la consultas MySQL Query Browser.

La primera es una herramienta muy simple y muy intuitiva que nos permite realizar diferentes cambios en la BBDD como, por ejemplo, creación de nuevas tablas, modificación de las tablas, gestión y administración de los usuarios, etc., de una forma muy rápida. Otra de las ventajas de esta herramienta es que permite realizar todas estas operaciones sin introducir el código SQL. Es una buena opción para las personas que solo están empezando a aprender SQL.

Ilustración 6: Vista entrada de MySQL Administrator

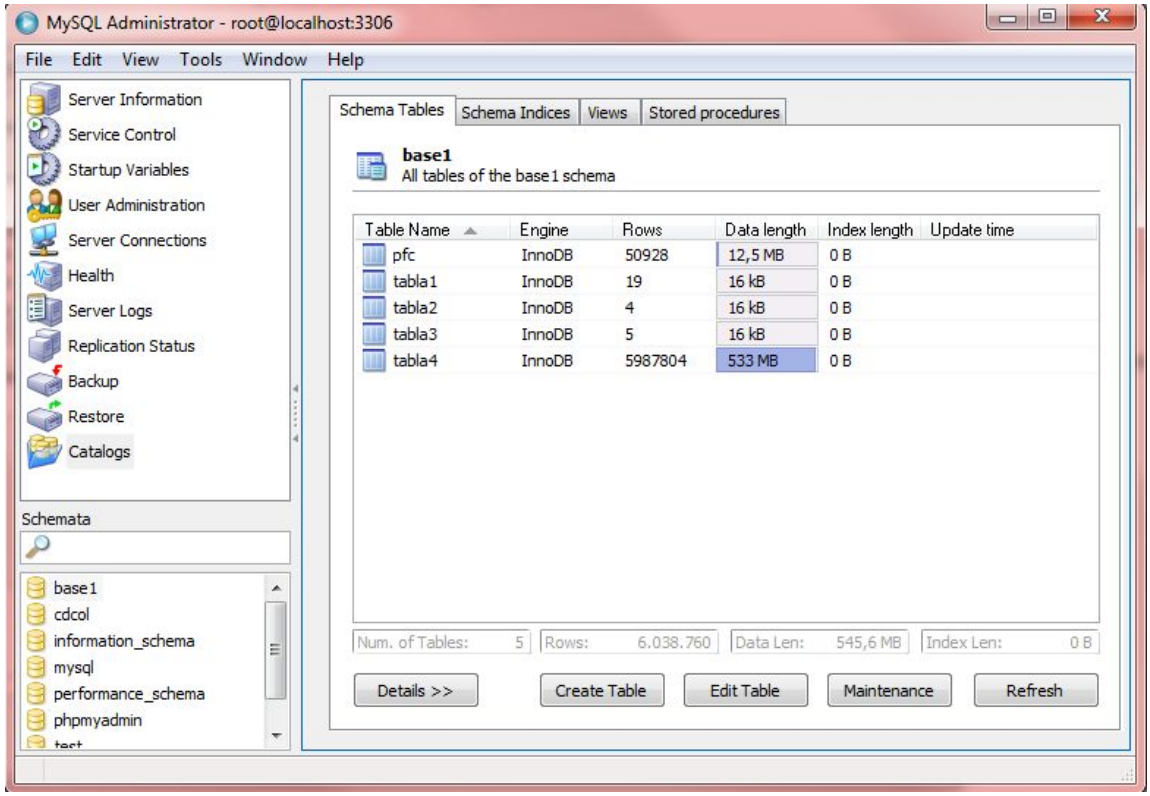

Ilustración 7: Catálogo de Bases y Tablas de MySQL Administrator

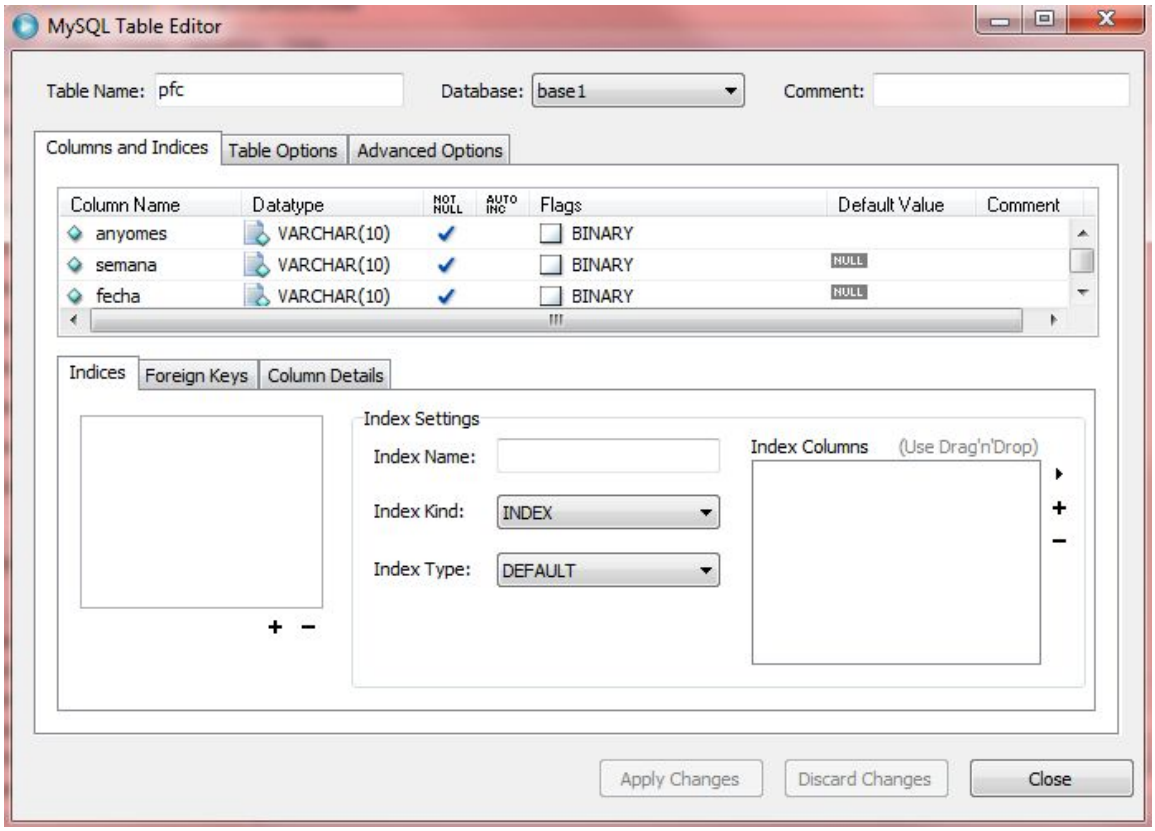

Ilustración 8: Editor de tablas de MySQL Administrator

Herramienta de consultas Query Browser, igual que la anterior, es muy útil y muy intuitiva. Es una herramienta imprescindible para ir probando las consultas que estamos diseñando.

Ilustración 9: Vista entrada de MySQL Query Browser

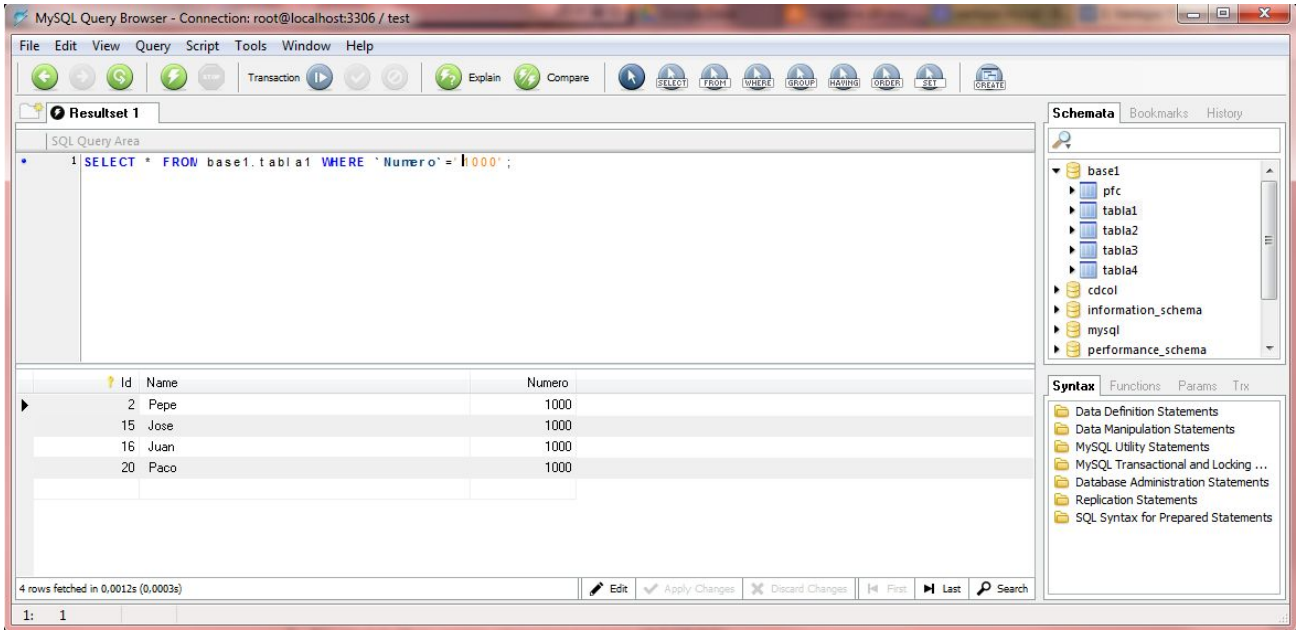

Ilustración 10: Diseño y ejecución de la consulta MySQL Query Browser

Una vez hecha esta pequeña introducción a MySQL y a las herramientas de la misma que iremos utilizando a lo largo del proyecto sigo con la explicación de la arquitectura del sistema.

Como el ***proceso 1*** dentro de la Fase 1 del primer bloque tenemos el "1. Diseño y configuración de una BBDD en el MySQL Server". Como su propio nombre indica, es un proceso para dejar nuestra BBDD en condiciones para poder trabajar con ella. Es un proceso que se realiza una solo vez, es decir no se repite cada vez que el usuario usa la herramienta. Consiste básicamente en la creación de una BBDD y en la creación y el diseño de las tablas que esta va a contener. También en este punto se crean los usuarios con las claves correspondientes y los privilegios. Estos parámetros, usuario y clave, son imprescindibles para usar la herramienta ya que cada consulta a la BBDD necesita establecer una conexión una conexión para la cual estos parámetros son imprescindibles.

Después de este proceso, cuando nuestra BBDD está lista para trabajar, se pasa al ***proceso 2*** relativo a la carga y la actualización periódica de los datos. Después de cargar el primer fichero con los datos a nuestra tabla, habrá que ir actualizándola cada mes, añadiendo un fichero nuevo con los datos del mes pasado. Este proceso se realizará a mediados de cada mes sin poder tener una fecha fija, ya que la generación y procesamiento de estos datos los realiza otro departamento. Por ello, son posibles los retrasos, lo que hace imposible automatizar este proceso totalmente haciendo que la carga se realice de una manera automática en un fecha y hora determinadas. Sin embargo, para hacer el proceso de carga y de actualización de datos más rápido he creado un pequeño script en Python que realiza esta tarea. El código dentro del script permite realizar la conexión a nuestra BBDD con el nombre de usuario y la clave contenidos en él:

```
1  import MySQLdb
2  print "Hola empezamos realizando la conexion a la BBDD"
3  try:
4      conexion = MySQLdb.connect(host = "localhost", user = "root", passwd = "******", db = "test1")
5      print "Conexion realizada correctamente"
6
7  except:
8       print "Error en el proceso de conexion. Revisa parametros y conectividad"
9
```

Ilustración 11: Script de carga y actualización de datos (Conexión)

Después de la conexión a la BBDD creamos el cursor de la conexión y ejecutamos las consultas que son necesarias insertándose dentro del código Python:

```
9
10  cursor=conexion.cursor() #Cursor de la conexion
11
12  try:
13      cursor.execute("SENTENCIA EN SQL PARA CARGAR DATOS")
14      ...
15
16  except:
17      print "Error en la consulta"
18
```

Ilustración 12: Script de carga y actualización de datos (Consultas)

La instrucción de MySQL utilizada para la carga de los datos es la LOAD DATA INFILE:

13.2.6 LOAD DATA INFILE Syntax

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
    [IGNORE number LINES]
    [(col_name_or_user_var,...)]
    [SET col_name = expr,...]
```

Ilustración 13: Sentencia LOAD DATA INFILE (imagen cogida desde el: https://dev.mysql.com/doc/refman/5.1/en/load-data.html)

Como se puede ver esta función tienen muchos parámetros que permiten ajustar el proceso de carga a las necesidades que tengamos. Básicamente lo que hace la función es leer los datos desde el fichero que proporcionamos y los coloca en la tabla que indicamos en la sentencia. Los aspectos quizás más importantes de esta sentencia que tenemos que tener en cuenta son el campo *[LOCAL]* y los campos *FIELDS TERMINATED BY 'String' y LINES TERMINATED BY 'String'*.

Si ponemos el campo *[LOCAL]*, se entiende como local la máquina del cliente final, por lo que el fichero será leído en el ordenador del cliente y se enviará al servidor. En nuestro caso, de

momento, no tenemos que utilizar esta opción ya que el fichero que tenemos que leer esta directamente en la máquina del servidor. Otro punto aquí a tener en cuenta es que el fichero a leer debe estar en el mismo directorio que nuestra BBDD, esto se hace por motivos de seguridad.

Otros dos campos citados arriba son para especificar las separaciones en el fichero que vamos a cargar, *FIELDS* para separar campos, es decir especificar cuándo termina el dato de cada columna, y *LINES* es para especificar cuándo se termina una línea y empieza la siguiente. Es muy importante especificar bien estos dos campos ya que de esto depende si nuestros datos serán cargados correctamente.

Una vez hemos hecho todo lo necesarios dentro de la nuestra conexión, tenemos que cerrar el cursor y la conexión:

```
25    cursor.close()
26    conexion.close()
```

Ilustración 14: Script de carga y actualización de datos (Cerramos el curso y la conexión)

Una vez que tenemos la BBDD preparada para el trabajo pasamos al Bloque2 de la Interfaz web. Este bloque, al igual que el otro, empieza con el diseño de la interfaz y su configuración. En la arquitectura lo tenemos marcado como ***proceso 3***, "3. Diseño inicial de la interfaz web y configuraciones necesarias". El diseño se iba transformando a lo largo del proyecto según íbamos introduciendo nuevos cambios y según íbamos avanzando en el proyecto. El diseño de la interfaz empieza en la Fase 2 y se modifica a lo largo de la Fase 3 y la Fase 4.
Una vez que tenemos nuestra página con una estructura inicial, podemos pasar a la configuración de la conexión entre la interfaz web y nuestra BBDD diseñada anteriormente. Esta conexión la realizamos en el ***proceso 4***, "4. Configuración de la conexión entre Bloque 1 y Bloque 2". Este proceso es muy importante, ya que como decía en los capítulos anteriores, la conexión la tendremos que establecer cada vez que vayamos a lanzar una consulta SQL a nuestra BBDD. La herramienta "Dreamweaver" en la que diseñamos la interfaz web permite configurar las conexiones y guardar dichas configuraciones, permitiendo de este modo utilizarlas directamente en las consultas sin necesidad de introducir los datos de conexiones cada vez que realizamos la consulta. (McFarland y Grover 2014)

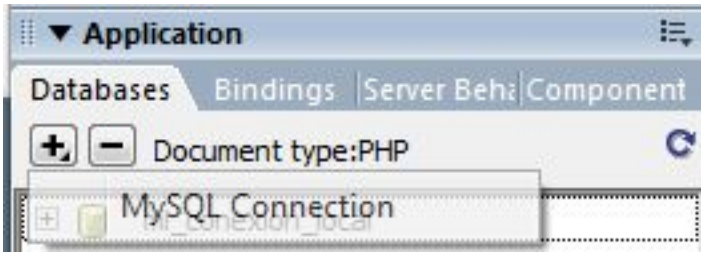

Ilustración 15: Configuración de la conexión a la BBDD (Dreamweaver)

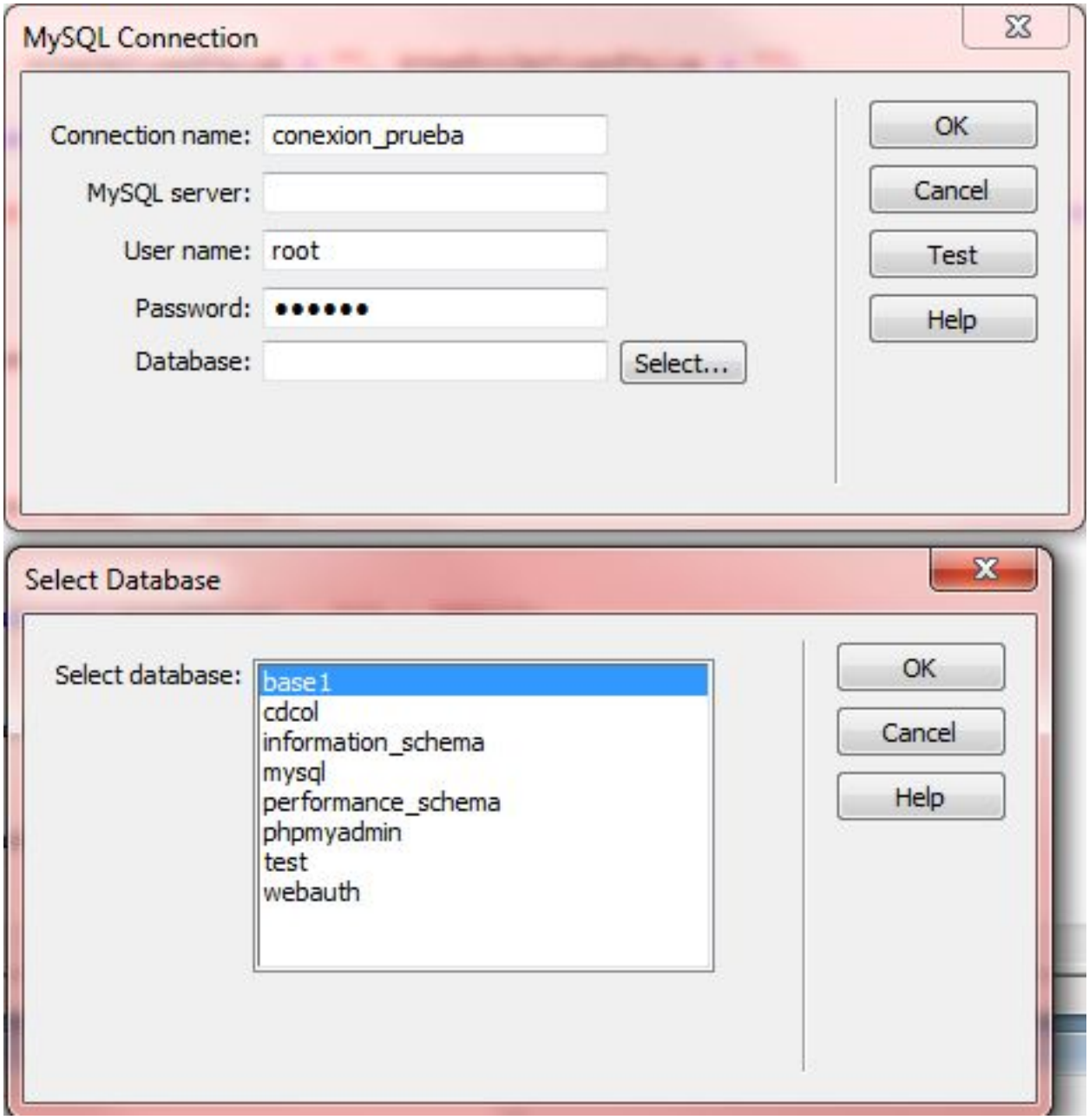

Ilustración 16: Configuración de la conexión a la BBDD (Dreamweaver)

```
//////////////////////////////   CONEXION A LA BASE   //////////////////////////////
mysql_select_db($database_mi_conexion_local, $mi_conexion_local);
```

Ilustración 17: Estructura PHP para establecer la conexión a la BBDD

Parámetros:

- *$database_mi_conexion_local:* nombre de BBDD a seleccionar.
- *$mi_conexion_local:* nombre de la conexión a la BBDD que hemos definido anteriormente.

Una vez llegado a este punto podemos empezar a diseñar las consultas SQL según los datos que necesitamos para representar nuestras gráficas, este proceso se realiza en el ***proceso 5*** "5. Diseño de las consultas SQL y su integración con el código PHP". Para ir probando dichas consultas a medida que las estamos diseñando y los datos que esas devuelven es muy cómodo utilizar la herramienta antes mencionada *MySQL Query Browser* o entrando desde el navegador a *phpmyadmin* introduciendo en él nuestro caso en el navegador *http://localhost:8080/phpmyadmin/.*

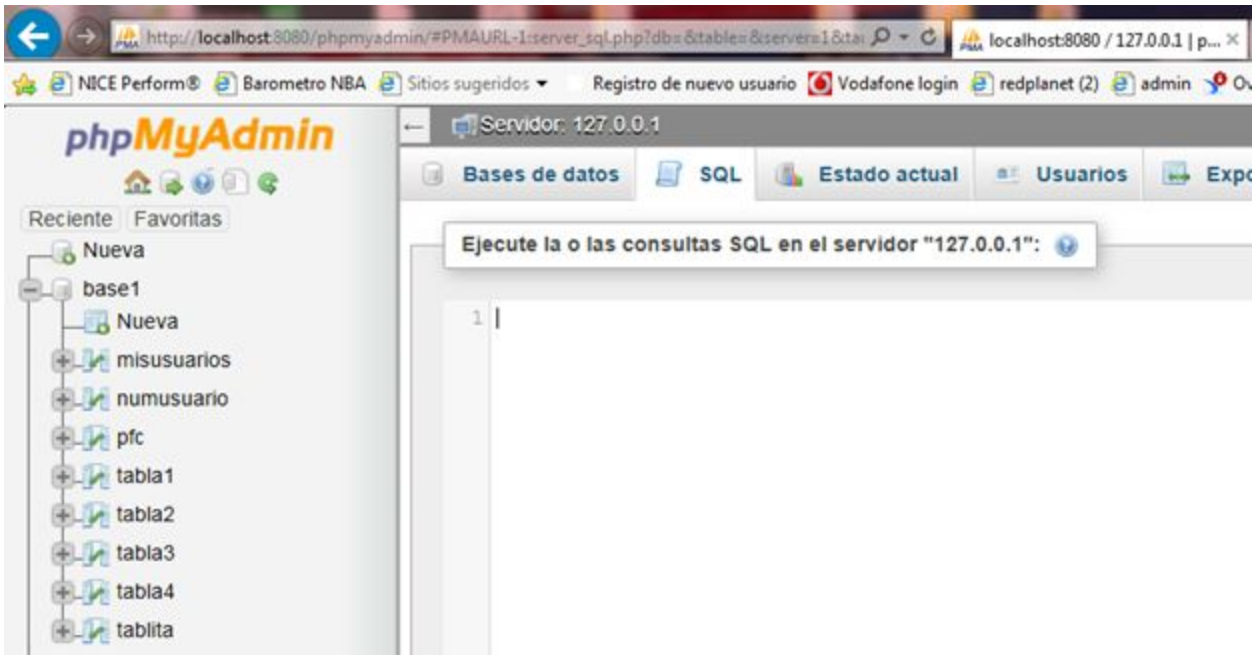

Ilustración 18: phpmyadmin Ventana principal

Una vez que tenemos nuestras primeras consultas, para probar cómo realmente funcionan dentro del sistema, las tenemos que integrar en la interfaz web integrándolas dentro del código PHP que permite esta opción mediante las funciones internas. La estructura es la siguiente:

```
$detalletabla = mysql_query($query_detalletabla, $mi_conexion_local) or die(mysql_error());
```

Ilustración 19: Estructura PHP para lanzar consulta SQL a nuestra BBDD

La estructura consiste de siguientes de siguientes campos:

- *$detalletabla*: es una referencia a un recurso externo que en nuestro caso son los datos solicitados (como si fuese un puntero).
- *$query_detalletabla:* es una variable en la que anteriormente definimos la consulta SQL a lanzar. La consulta se escribe dentro de comillas (*""*) y sin poner el punto y coma al final.
- *$mi_conexion_local:* es el nombre de la conexión a la BBDD que hemos definido anteriormente.

En este punto entramos en el ***proceso 6*** “6.Envio de consultas MySQL hacia nuestra BBDD” y en él ***Proceso 7*** “7.Devolucion de los datos correspondientes a las consultas enviadas”, tras lanzar la consulta. Si no existe ningún error de sintaxis o de conexión, la BBDD devolverá los datos en forma de un puntero que realmente apunta a los mismos. Para poder trabajar con estos, sea imprimirlos, verlos o procesarlos, los tenemos que leer desde el puntero teniendo en cuenta la estructura que tienen.

En el momento en el que tras lanzar la consulta a nuestra BBDD tenemos el puntero que apunta a los datos de interés, entramos en el ***proceso 8*** “8.Procesado de los datos y su representación en forma de gráficas”.
Para empezar a manejar los datos, tenemos que leerlos desde el puntero que nos ha devuelto nuestra consulta. Para hacer este proceso se utiliza la función *mysql_fetch_assoc(),* que utilizamos de siguiente modo, pongo un ejemplo:

```
////////////////////////////////////////////////////
$names = array();
$numeros = array();
$pos = 0;
// leyendo datos
while ($row = mysql_fetch_assoc($detalletabla)) {
    $names[$pos] =  $row['Name'];
    $numeros[$pos] = $row['Numero'];
    $pos = $pos + 1;
}
```

Ilustración 20: Leyendo datos devueltos (mysql_fetch_assoc)

Mientras en nuestro puntero exista una fila con datos, leemos esta fila y vamos colocando los campos correspondientes a diferentes columnas (en ejemplo: *Name y Número*) en los nuevos arrays correspondientes que iremos manejando en el procesamiento. Con esta acción pretendemos separar los datos en los arrays independientes según las columnas en las que aparecen dentro de la BBDD, guardando la relación de posiciones entre dichas columnas.

Una vez que hemos leído nuestros datos desde el puntero podemos empezar con su representación. Aquí cabe destacar que en función de las gráficas que queremos mostrar podríamos pasar los datos leídos directamente como parámetros de las gráficas o podemos procesar estos datos previamente y ya posteriormente pasarlos como parámetros. Este procesamiento adicional será necesario en algunos tipos de gráficas y será descrito con más detalle en el Capítulo de Diseño. Principalmente son acciones para ajustar los casos en los que un parámetro a medir no tiene datos de unos meses, sin embargo otros parámetros de la misma gráfica si los tiene, en estos casos tenemos que detectar estos meses y rellenarlos con

los ceros para los valores que pasamos a las gráficas sean realmente correspondientes con los meses.

Ahora bien, una vez estamos en el punto de mostrar las gráficas tenemos que elegir una de las múltiples opciones que tenemos para este fin. (*Google Chart, pChart, PlotKit, Vertical Bar Graphs, Real World Bar Graphs,..*). Finalmente elegí la opción de utilizar pChart, básicamente por dos motivos, por la sencillez en la utilización y por la variedad de las gráficas que nos ofrece. La sencillez consiste en que el bloque de código correspondiente a la representación se inserta directamente dentro del código PHP sin necesidad de utilizar ninguna herramienta o acción externa (en el caso de *Google Chart API* tenemos que hacer llamadas externas a la aplicación de *google*). Solo tenemos que añadir dos sentencias de *include* para incluir las librerías de pChart:

```
include("C:/xampp/htdocs/allview/pChart/pData.class");
include("C:/xampp/htdocs/allview/pChart/pChart.class");
```

Ilustración 21: Sentencias include de pChart

Después de leer nuestros datos desde el puntero, o adicionalmente procesarlos en el caso necesario, pasamos nuestros arrays de datos como parámetros al código de pChart. El formato de este paso de datos depende del tipo de gráfica que vamos a mostrar y del número de series que queremos representar:

```
/////GRAFICA/////
 // Dataset definition
 $DataSet = new pData;
 $DataSet->AddPoint($recos,"RECO");//Serie 1
 $DataSet->AddPoint($stops,"STOP");//Serie 2
 $DataSet->AddPoint($outs,"OUTOFRECO");//Serie 3
 $DataSet->AddPoint($vaciosreal,"VACIOS_PILOTO");//Serie 4
 //$DataSet->AddPoint($names,"Names");
 //$DataSet->ImportFromCSV("Sample/bulkdata.csv",",",array(1,2,3),FALSE,0);
 $DataSet->AddAllSeries();
 $DataSet->SetAbsciseLabelSerie($ejesemana);//ponemos array de semanas como eje
 $DataSet->SetYAxisName("#Cambios Tarifa");
 // Initialise the graph
```

Ilustración 22: Código gráficas. Pasamos datos a pChart

Siguientes líneas dentro del código de pChart permiten ajustar diferentes parámetros de las gráficas (títulos, fondos, estilos, tamaños, etc.) para dejarlas con el diseño que queremos:

```
// Initialise the graph
$Test = new pChart(700,230);
$Test->setFontProperties("Fonts/tahoma.ttf",8);
$Test->setGraphArea(70,30,680,200);
$Test->drawFilledRoundedRectangle(7,7,693,223,5,240,240,240);
$Test->drawRoundedRectangle(5,5,695,225,5,230,230,230);
$Test->drawGraphArea(255,255,255,TRUE);
$Test->drawScale($DataSet->GetData(),$DataSet->GetDataDescription(),SCALE_NORMAL,150,150,150,TRUE,0,2);
$Test->drawGrid(4,TRUE,230,230,230,50);
```

Ilustración 23: Código gráficas. Inicialización del gráfica

Una vez definido el diseño de la gráfica e inicializado el mismo procedemos a pintarlo y finalmente a guardarlo en el directorio especificado:

```
// Draw the 0 line
$Test->setFontProperties("Fonts/tahoma.ttf",6);
$Test->drawTreshold(0,143,55,72,TRUE,TRUE);
// Draw the line graph
$Test->drawLineGraph($DataSet->GetData(),$DataSet->GetDataDescription());
$Test->drawPlotGraph($DataSet->GetData(),$DataSet->GetDataDescription(),3,2,255,255,255);
// Finish the graph
$Test->setFontProperties("Fonts/tahoma.ttf",8);
$Test->drawLegend(596,150,$DataSet->GetDataDescription(),255,255,255);
$Test->setFontProperties("Fonts/tahoma.ttf",10);
$Test->drawTitle(50,22,"Cambios_Tarifa Recomendador",50,50,50,585);
//Lo guardo en directorio de allview
$Test->Render("C:/xampp/htdocs/allview/imagenes/query1.png");
```

Ilustración 24: Código gráficas. Dibujamos y guardamos el gráfica

Una vez guardadas las gráficas en el nuestro directorio, serán leídos desde allí para ser representadas en la interfaz web. Esta parte está incluida dentro del código HTML. Por lo que cada vez que recargamos la página se refresca la imagen guardada en el directorio y por lo que se refresca la imagen en la interfaz:

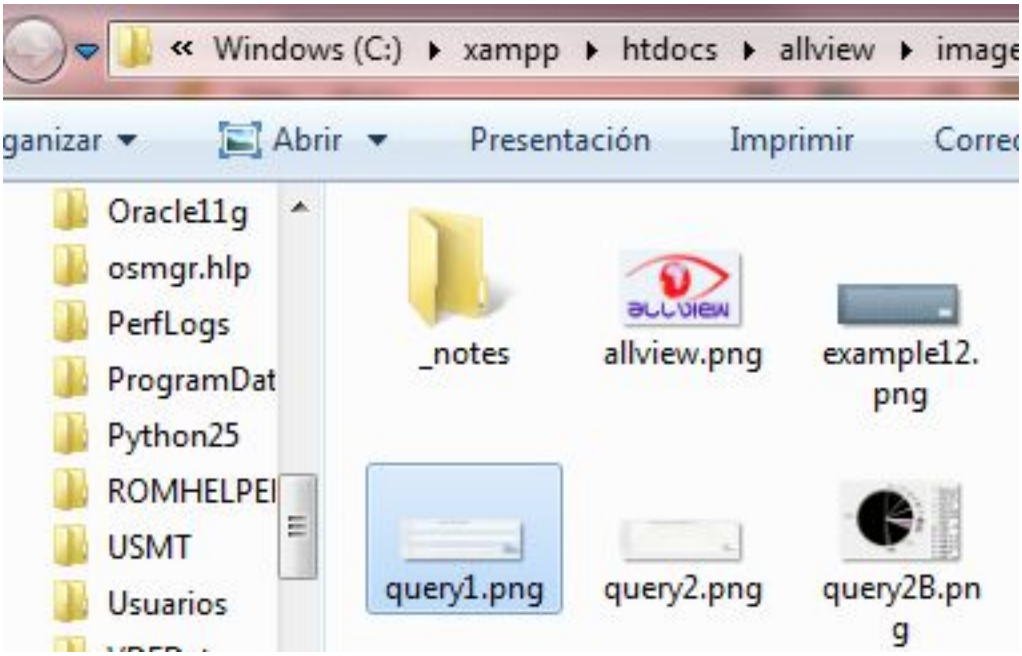

Ilustración 25: Gráfica guardada en el directorio especificado

```
</tr>
    </table>
  </div>
  <div id="Content">
    <div>
      <table width="1019" height="39" border="0">
        <tr>
          <td width="24%"><div align="center"><img src="imagenes/query1.png" alt="query1" width="700" height="230" /><button
onclick="location.reload()" style="vertical-align:middle">Reload</button></div></td>
          <td width="26%" rowspan="4"> </td>
          </tr>
          <tr>
```

Ilustración 26: Código gráficas. Código HTML

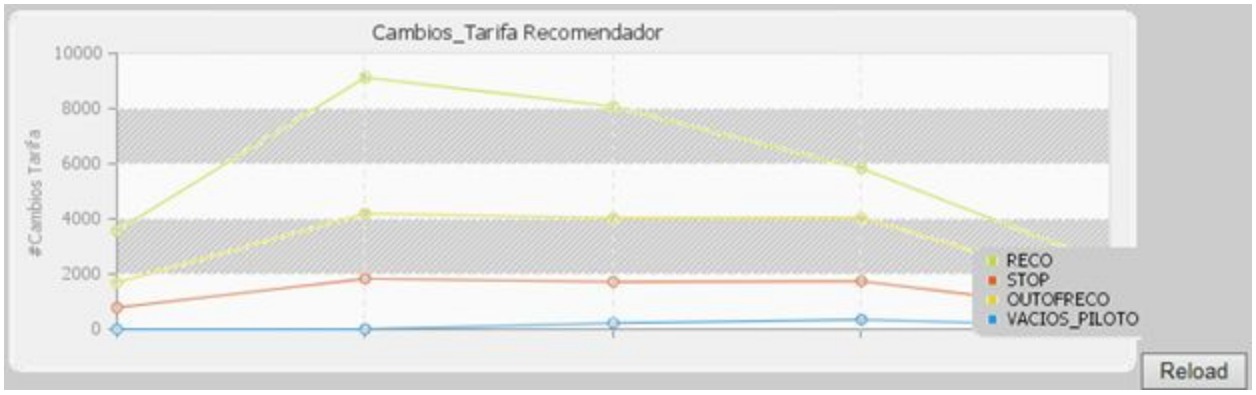

Ilustración 27: Gráfica dentro de la interfaz

De este modo queda descrito el proceso completo del funcionamiento de nuestra herramienta, empezando por la carga de datos a la BBDD y terminando con la visualización de los resultados por parte del usuario. Realmente el usuario no se entera de todos los procesos que se están ejecutándose dentro de la herramienta ya que interactúa directamente con el *FrontEnd*, es decir con la interfaz web.

5.- Diseño

Diseño a más bajo nivel de los módulos de tu sistema (clases principales, métodos principales):

- Modelo de datos de cada base de datos
- Diseño interno de cada módulo
- Vistas del front-end web

En este bloque se explica en más detalle el diseño de la BBDD, sobre todo el formato de datos y su significado, y por otro lado se presenta más detalladamente en que consiste la interfaz web.

5.1. Diseño del formato de datos

Empezaré explicando por el formato de datos que tenemos almacenados en nuestra BBDD. En este punto cabe destacar que todos los procesos de cruce de datos entre varias tablas para conseguir un formato deseado y descartar los datos que no interesan se realizan fuera de nuestra BBDD. Para ser más preciso, se realizan algunas simplificaciones de la tabla original del departamento que nos proporciona la información. Posteriormente en nuestro equipo hacemos cierto cruce de la tabla que nos pasan con una tabla maestra que tenemos. Este paso lo realizamos para traducir ciertos códigos que se utilizan en otros departamentos a los nombres que se utilizan en nuestro equipo, y también para hacer una agrupación de ciertos parámetros.

En concreto traducimos los códigos que tienen asignadas las tarifas a nombres de tarifas reales. Y por otro lado agrupamos diferentes conceptos de cambios de tarifa agrupándolos por clases que manejamos en nuestro equipo.

Estas operaciones las realizamos a través de la herramienta *Access* para el manejo de BBDD. De este modo, tras realizar todas estas operaciones tenemos un fichero mucho más reducido, es decir, solo con los datos que nos interesan y con el formato que nos interesa. Todo esto permite evitar cruces de datos innecesarios de varias tablas dentro de nuestra BBDD MySQL, aumentando de este modo la velocidad de respuesta y simplificando el diseño de las consultas y de la BBDD en sí.

A continuación presento el proceso de “preprocesado” de datos para obtener una tabla con el formato para cargarla a nuestra BBDD:

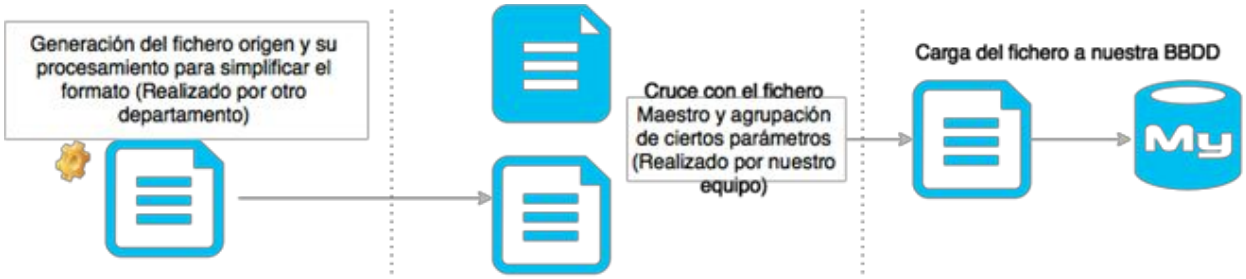

Ilustración 28: Preprocesado del fichero origen

Una vez descrito este proceso de preprocesado el fichero, empiezo a describir el fichero en sí, describiendo básicamente qué datos contiene y de qué columnas está compuesto.

La principal y la única tabla que en este momento utilizamos en nuestra BBDD es la tabla que nos permite obtener la información necesaria para entender qué está sucediendo con los cambios de tarifas a nivel de agencia y con cada una de las tarifas en concreto. También podemos monitorizar que está sucediendo con diferentes parejas de tarifas (*tarifa_origen – tarifa_destino*).

La tabla contiene dieciocho columnas, que tienen siguientes nombres y significados:

Nombre de Columna	Que representan los datos de columna
1. Anyomes	Representa el año y el mes de la operación de cambio de tarifa.
2. Semana	Representa la semana en la que fue efectuado el cambio
3. Fecha	Representa la fecha en la que fue efectuado el cambio en formato dd/mm/aaaa
4. Msisdn	Representa el msisdn (*Mobile Station Integrated Services Digital Network*) o lo que es lo mismo

	que el número de teléfono del cliente que realiza el cambio.
5. Tipomigra	Específica a que tipo pertenece el cambio de tarifa, o mejor decir a que se debe este cambio (cambio de tarifa simple, canje, etc.)
6. Meses_para_exp_tarifa	Especifica cuantos meses le quedan de permanencia en la tarifa a un cliente en el caso de que tenga permanencia en la tarifa
7. Meses_para_exp_cp_vdf	Especifica cuantos meses le quedan al cliente de permanencia en Vodafone en el caso de que tenga la permanencia.
8. Tarifa_origen	Especifica la tarifa origen del cliente en el momento de realizar el cambio de tarifa
9. Tarifa_destino	Especifica la tarifa destino que tiene el cliente tras realizar el cambio de tarifa
10. Healthy	Es una especificación que el sistema asigna a este cambio de tarifa, en concreto a la pareja de *tarifa_origen – tarifa_destino.* Depende sobre todo de

		tarifa_destino a partir de una específica tarifa_origen y del segmento del cliente. En función de esto el sistema asigna el grado de como es recomendable realizar este cambio.
11.	Healthyagrupado	Especifica la agrupación del parámetro anterior. Este campo se obtiene haciendo un cruce con el fichero maestro
12.	Centro	Especifica el centro (Centro de atención de las llamadas) en el que fue efectuado el cambio de tarifa
13.	Departamento	Especifica el departamento del centro en el que fue efectuado el cambio de tarifa
14.	GT	Especifica el grupo de trabajo del departamento en el que fue realizado el cambio de tarifa
15.	Agente	Especifica el login del agente que realizó el cambio de tarifa
16.	Contador	Es un valor adicional puesto a uno (1). Se utiliza como contador en algunas operaciones que se

		realizan fuera de esta herramienta
17.	NIF	Es el número identificador del cliente
18.	Campanya	Específica en qué tipo de campaña fue realizado el cambio. El cambio podría ser realizado en diferentes campañas (a través de una NBA, entrante, saliente, prevención, etc.)

Ilustración 29: Contenidos de tabla en BBDD

Con los datos descritos en la tabla anterior podemos monitorizar de manera periódica o puntual muchos comportamientos, empezando por el comportamiento de una pareja de tarifas y terminando por el comportamiento de un centro o un agente en concreto. En nuestro caso, y en la situación actual, nos interesa monitorizar el comportamiento general de los centros y el comportamiento de cambios de tarifa con una tarifa origen especial. De este modo en función de las necesidades que tengamos para monitorizar algún dato de manera periódica, podremos añadir esta monitorización en nuestra interfaz web, modificando alguna de las consultas (por ejemplo si necesitamos ver el comportamiento de otra tarifa_origen) o copiando el bloque de consulta y representación, y cambiando algunos de los parámetros de las consultas.

5.2. Diseño de la interfaz web

Para implementar la interfaz web en PHP y HTML, he utilizado la herramienta Dreamweaver, que permite diseño de las interfaces de este tipo de una forma muy cómoda y rápida. La herramienta permite realizar el diseño de forma gráfica, con el código o mezclando estos dos procedimientos según la necesidad que tengamos en cada momento del desarrollo, permitiendo de este modo realizar los cambios en un lado y verlos en el otro.

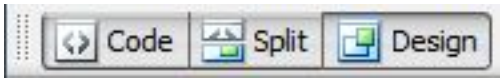

Ilustración 30: Tres modos de desarrollo en Dreamweaver

Además de esto, Dreamweaver permite muchas opciones que nos pueden facilitar la creación y el diseño de nuestra interfaz web. Una de estas opciones es la creación de las plantillas que podemos utilizar para el desarrollo de todas las páginas que contendrá nuestra interfaz. De este modo, si a lo largo del desarrollo de la interfaz nos surge la necesidad de realizar algún cambio o añadir algún elemento nuevo, solo tendríamos que realizar este cambio en nuestra plantilla y no en todas las páginas de nuestra interfaz. Este nos permite realizar cambios de manera eficiente y ahorrando el tiempo de desarrollo.

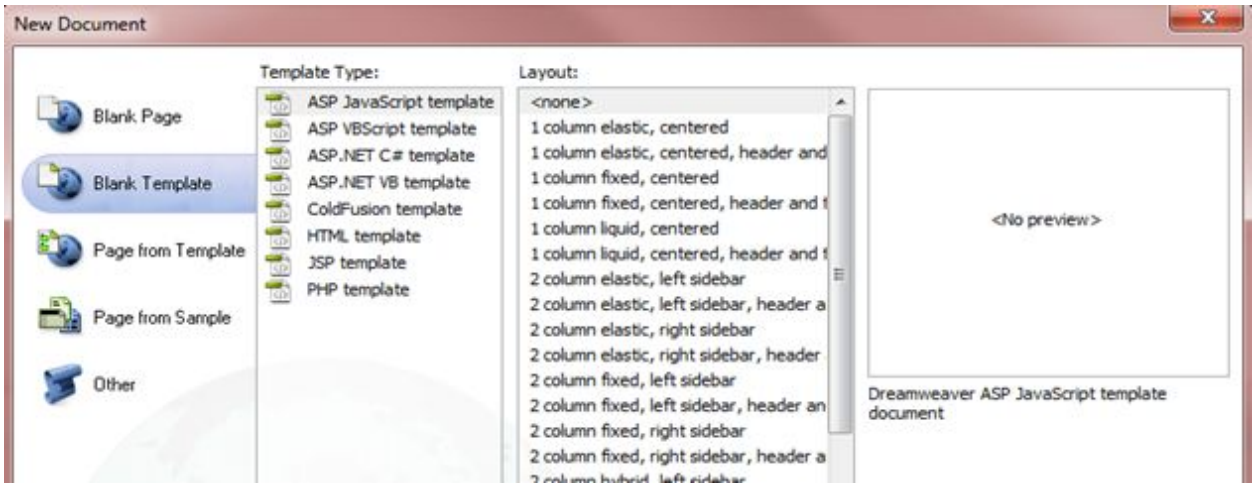

Ilustración 31: Selección del *template* para crear página nueva

Otra de las opciones muy útiles que nos permite realizar el Dreamweaver es el establecimiento de las conexiones con nuestras BBDD de una forma muy fácil e intuitiva. Además cuando en algún punto del desarrollo utilizamos nuestra conexión, el Dreamweaver incluirá el código necesario para realizar la misma en el lugar necesario. Sobre todo se ve en los casos cuando añadimos algunos diseños de formularios para realizar consulta de datos a nuestra BBDD o un formulario para añadir datos a la BBDD desde la interfaz web. En el siguiente ejemplo vemos como diseñamos en el modo *"Diseño"* un formulario para que el usuario pueda dejar su comentario en una tabla habilitada especialmente para este fin.

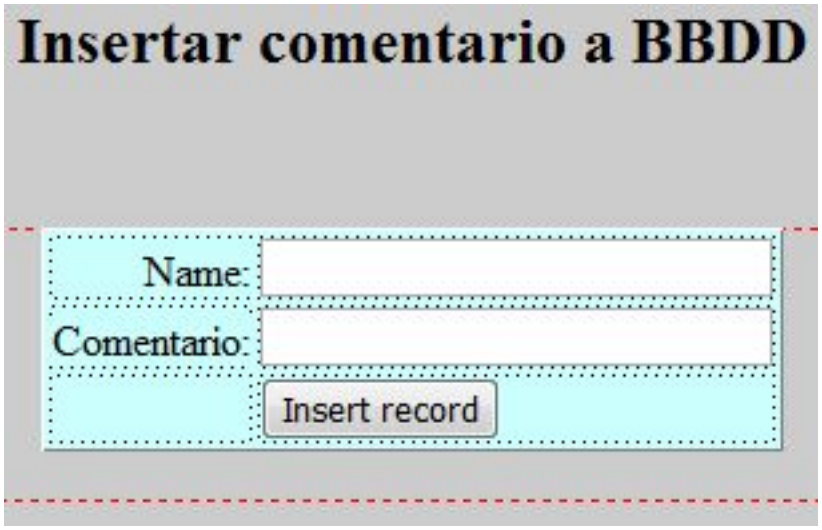

Ilustración 32: Diseño de formulario en modo "Diseño". Dreamweaver.

Y ahora vemos como el Dreamweaver añade el código correspondiente a este formulario en la parte de "Codigo".

```
<form action="<?php echo $editFormAction; ?>" method="post" name="form1" id="form1">
  <table align="center">
    <tr valign="baseline">
      <td nowrap="nowrap" align="right">Name:</td>
      <td><input type="text" name="Name" value="" size="32" /></td>
    </tr>
    <tr valign="baseline">
      <td nowrap="nowrap" align="right">Comentario:</td>
      <td><input type="text" name="Comentarioo" value="" size="32" /></td>
    </tr>
    <tr valign="baseline">
      <td nowrap="nowrap" align="right"> </td>
      <td><input type="submit" value="Insert record" /></td>
    </tr>
  </table>
  <input type="hidden" name="MM_insert" value="form1" />
</form>
```

Ilustración 33: Parte del código correspondiente al formulario. Dreamweaver.

Una vez hecha esta pequeña introducción de las opciones más interesantes que nos permite Dreamweaver, volvemos al diseño de nuestra interfaz web. Como ya he hablado en varias ocasiones, el objetivo es que el diseño de la interfaz web sea sencillo y que esta resulte fácil de usar.

Como en cualquier sitio web, tenemos una página inicial, llamada *index.php* en este proyecto. Es la primera página que vemos cuando abrimos nuestra interfaz web. Para acceder a este

sitio tenemos que introducir en el navegador siguiente dirección: *http://localhost:8080/allview/index.php*.

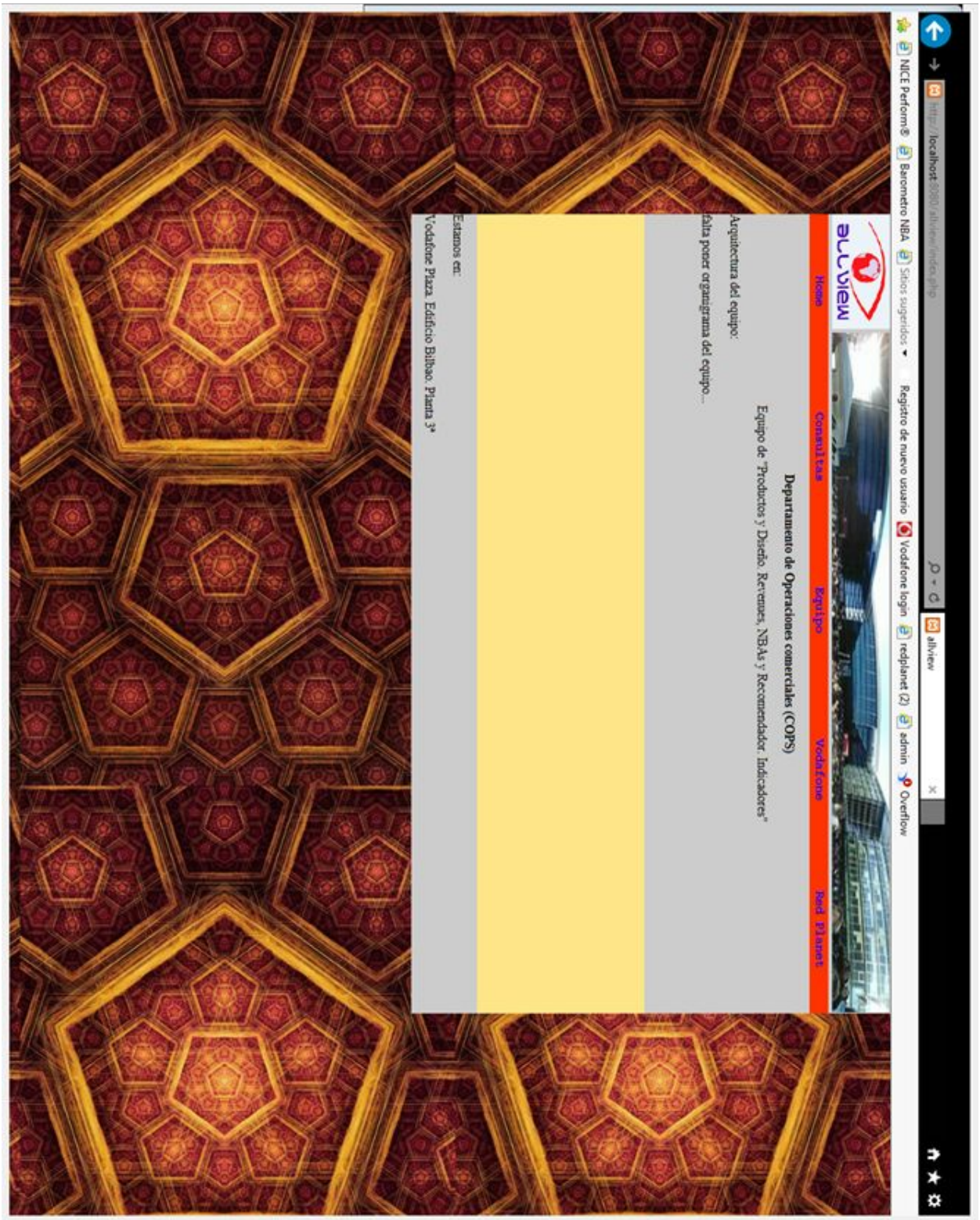

Ilustración 34: Interfaz web (index.php)

En esta página podemos ver una pequeña introducción al departamento y al equipo al que pertenezco con un organigrama y con los nombres de los miembros del mismo y con las responsabilidades que tienen asignadas. Esto ayudará a los usuarios a saber a qué miembro del equipo deben dirigirse para tratar sobre los temas diferentes que se gestionan dentro del equipo.
En la parte superior de la página (justo debajo de la imagen) vemos el panel de menú o panel de navegación con los sitios a los que podemos acceder a continuación. Aquí cabe destacar que este panel lo vemos siempre, independientemente de la página en que esté situado el usuario.
Empezaré explicando la primera opción empezando por la izquierda, la opción de "*Home*". Esta opción permite volver a la página principal (*index.php*) desde cualquier ventana en la que estemos. Cabe destacar que misma funcionalidad tiene la imagen en la parte superior izquierda (*allview*). Esta imagen, que representa el nombre que he dado a la herramienta, permite volver a la página inicial.

La siguiente opción que vemos es la opción de "*Consultas*". Como el propio nombre indica, es el sitio donde vamos a mostrar diferentes resultados de la actividad de los centros de atención telefónica. Accedemos a esta opción pinchando en "Consultas", o directamente introduciendo en el navegador *http://localhost:8080/allview/consultas.html*. Básicamente, en esta opción disponemos de cuatro posibles consultas y una opción para insertar el comentario "*Insertar el comentario*". Además, como he indicado antes, tenemos el panel de menú justo debajo de la imagen.

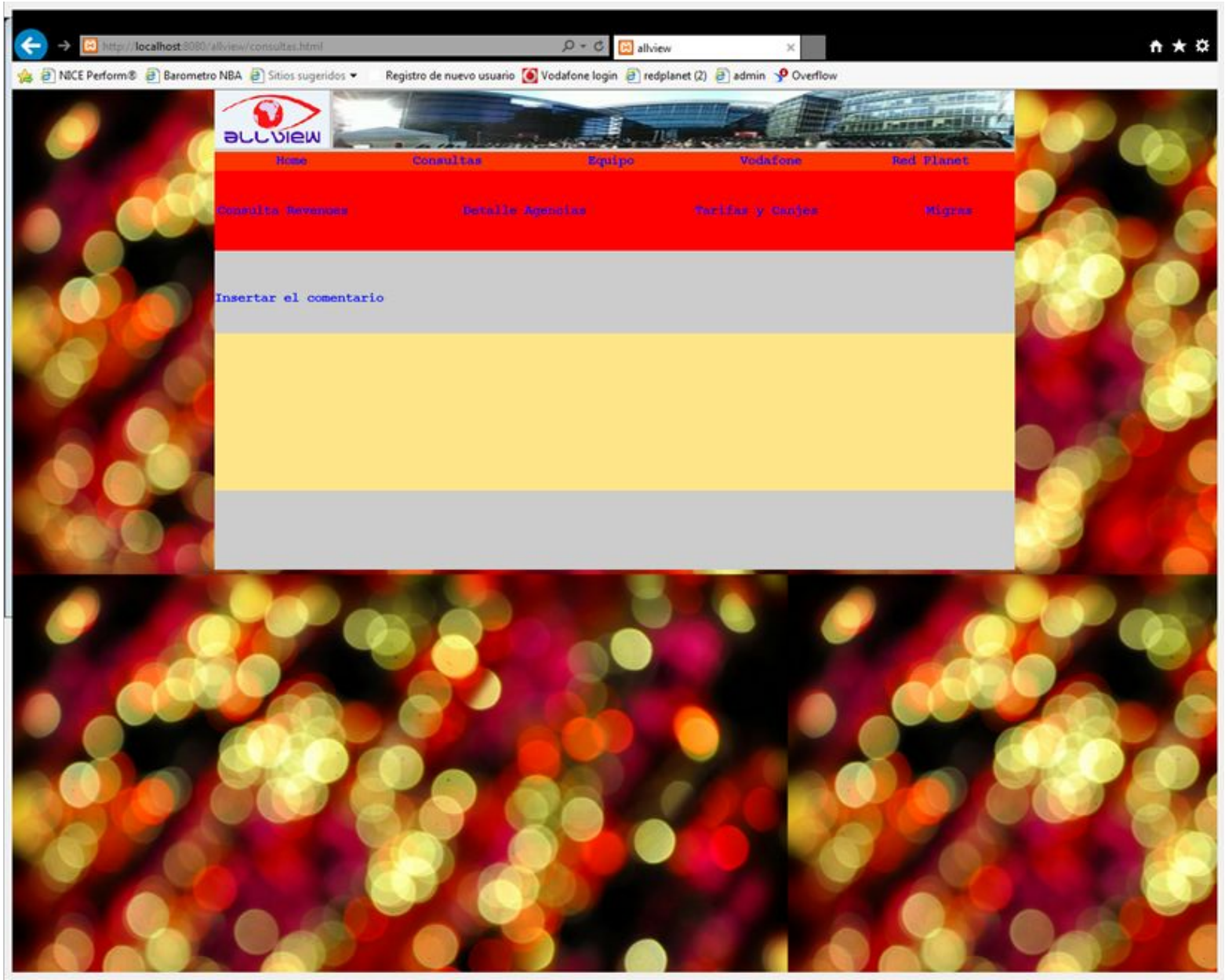

Ilustración 35: Interfaz web (consultas.html)

En este momento la primera opción, "*Consulta Revenues*", y la última, "*Migras*", no tienen ninguna información. Estas dos opciones se desarrollarán en el futuro en el caso en el que realmente surge la necesidad de ampliar la herramienta con estos datos.

En el caso de la "*Consulta Revenues*", no está incluida porque realmente estos datos se calculan y se muestran en un cuadro de mando de Excel sin la necesidad de procesar grandes volúmenes de datos, ya que ya nos vienen precalculados desde otros departamentos. Por lo que en el caso de que quisiéramos incluir esta información tendríamos que vincular de algún modo los datos del cuadro de mando sin necesidad de utilizar ninguna tabla de BBDD. Está pensada como una posible mejora para ampliar la información en la interfaz. Se puede decir que los ingresos (*revenues*) es el dato más global que se gestiona, y para entrar en más detalle se analizan otros indicadores.

La siguiente opción que tenemos es la de "*Detalle Agencias*", a la que accedemos pinchando en el propio icono o introduciendo en el navegador *http://localhost:8080/allview/detalle_agencias.php*. En esta opción, como el propio nombre indica, tenemos detalle de la actividad de los centros. Más en concreto, servicios específicos de los centros, referida a diferentes cambios de tarifa que se realizan. Esta opción nos permite analizar esta actividad desde diferentes puntos de vista que nos pueden interesar en cada momento, como por ejemplo tarifa origen, destino, tipo de migración, etc.
La actividad de cada uno de los centro es totalmente independiente de los demás centros, por lo que la forma de trabajo que se lleva a cabo en cada uno de ellos es diferente. Es debido a muchos factores diferentes como el tipo de agentes que trabajan en los centros, la formación que reciben, el foco principal de cada uno de los centros o simplemente la localización geográfica del centro. Todo esto se traduce en que algunos centros pueden llevar a cabo algunas actividades mejor que otros, y es muy importante disponer de esa información para saber cómo actuar en diferentes campañas comerciales que se llevan a cabo en cada momento.
Como ejemplo podemos pensar que si un mes nuestro objetivo es potenciar la venta de un *productoX*, y sabemos que históricamente el centro que mejor realiza la venta del *productoX* es por ejemplo *CentroA*, nuestra actividad principal durante esta campaña se pondría en el *CentroA*. Fuera de esta campaña, nuestra función sería analizar las causas de por qué el *productoX* no se vende igual de bien en los demás centros y plantear posibles estrategias para resolver este problema. Es principalmente por esto la información que veremos en el opción de "*Detalle Agencias*" tiene mucha importancia.

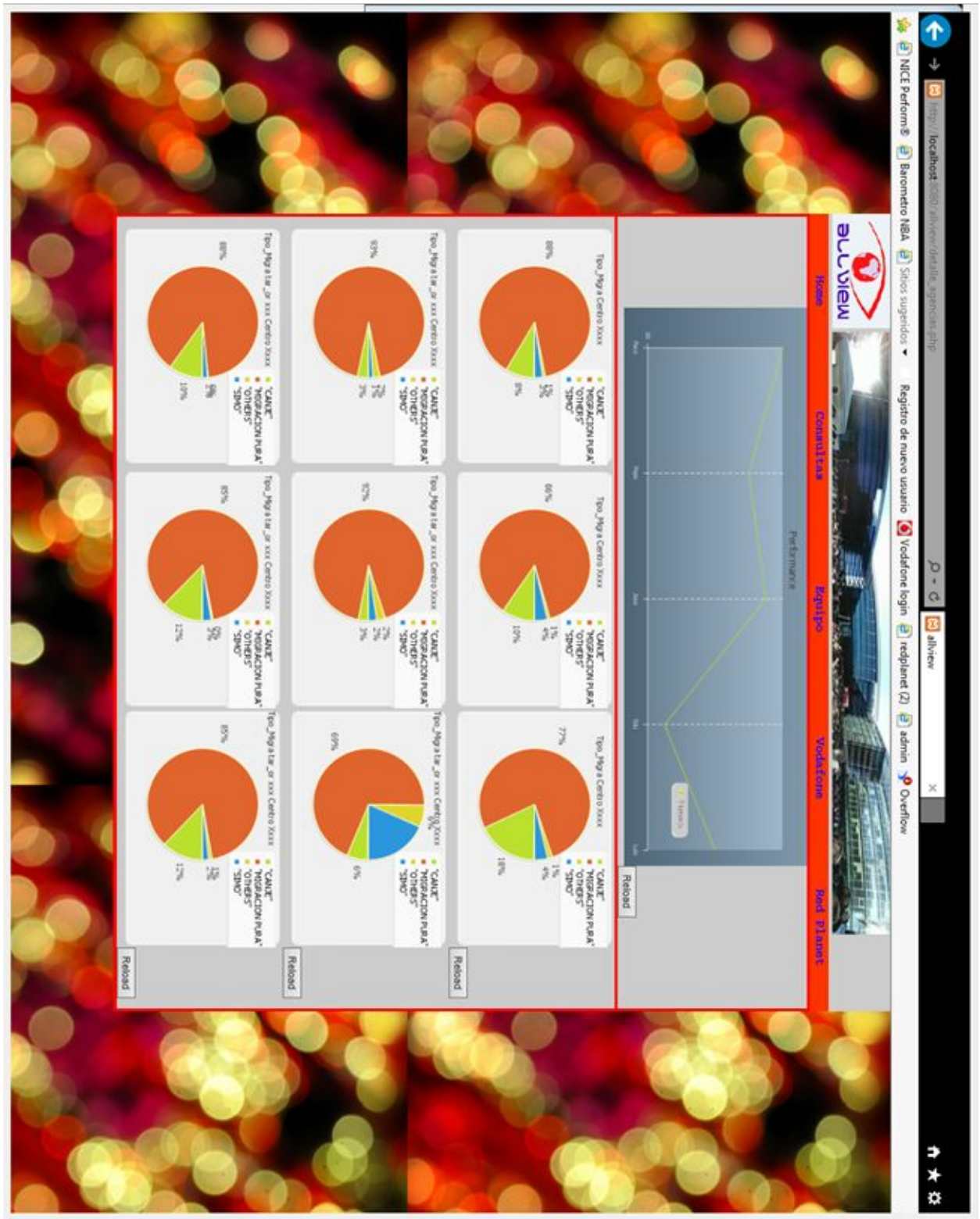

Ilustración 38: Interfaz web (Detalle Agencias)

La siguiente y última opción que nos queda por comentar dentro de las opciones de "*Consultas*" es la de "*Tarifas y Canjes*". Igual que antes, accedemos pinchando en la opción o introduciendo

en el navegador *http://localhost:8080/allview/tarifas_canjes.php*. En esta opción la idea es mostrar con más detalle cómo se realizan los cambios de tarifa, teniendo en cuenta tarifa origen, destino y diferentes parejas de mismos. Esta información, igual que la de “Detalle Agencias”, es muy importante y permite monitorizar cómo se comportan clientes con diferentes tarifas en función de aparición de nuevas tarifas, cambios de condiciones de tarifas antiguas, diferentes campañas puntuales, etc. Permite detectar los comportamientos de clientes que pertenecen a una tarifa específica, como a qué tarifa se tienden a cambiar más y a cuál menos, y en función de esto tomar medidas necesarias como podría ser desarrollo de campañas de promociones, descuentos, etc. La información en las gráficas se muestra con evolución semanal.

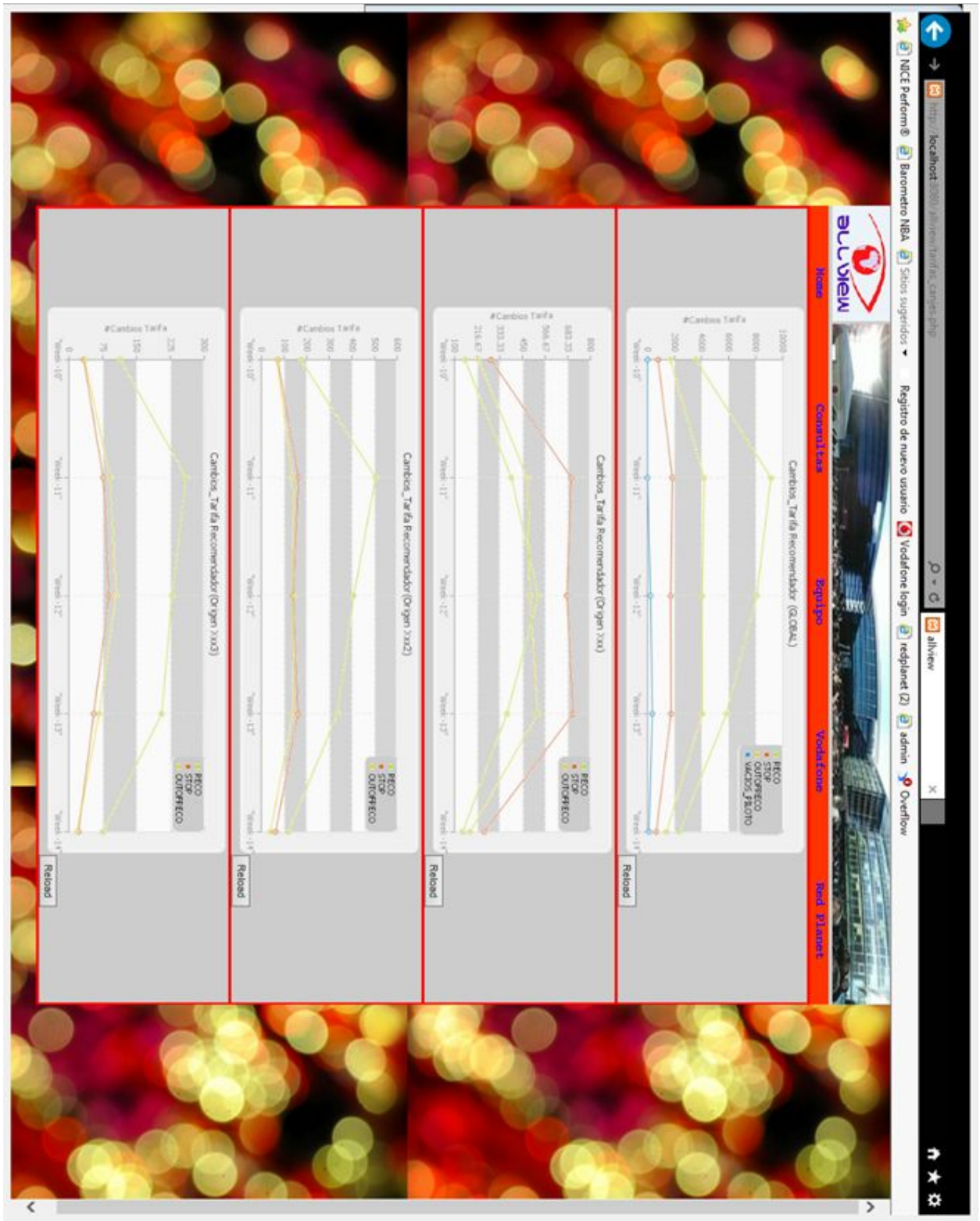

Ilustración 39: Interfaz web (Tarifas y Canjes)

La última opción que nos queda dentro de la opción de “Consultas” es “Insertar el comentario. Esta opción está pensada para que los usuarios de nuestra herramienta puedan dejar algún

comentario sobre la misma, indicando qué posibles cambios deberíamos realizar y qué posibles consultas o mejoras deberíamos añadir. Pinchando en esta opción aparece una ventana (*http://localhost:8080/allview/insertar_registro_tabla1.php*) en la que el usuario puede introducir un comentario.

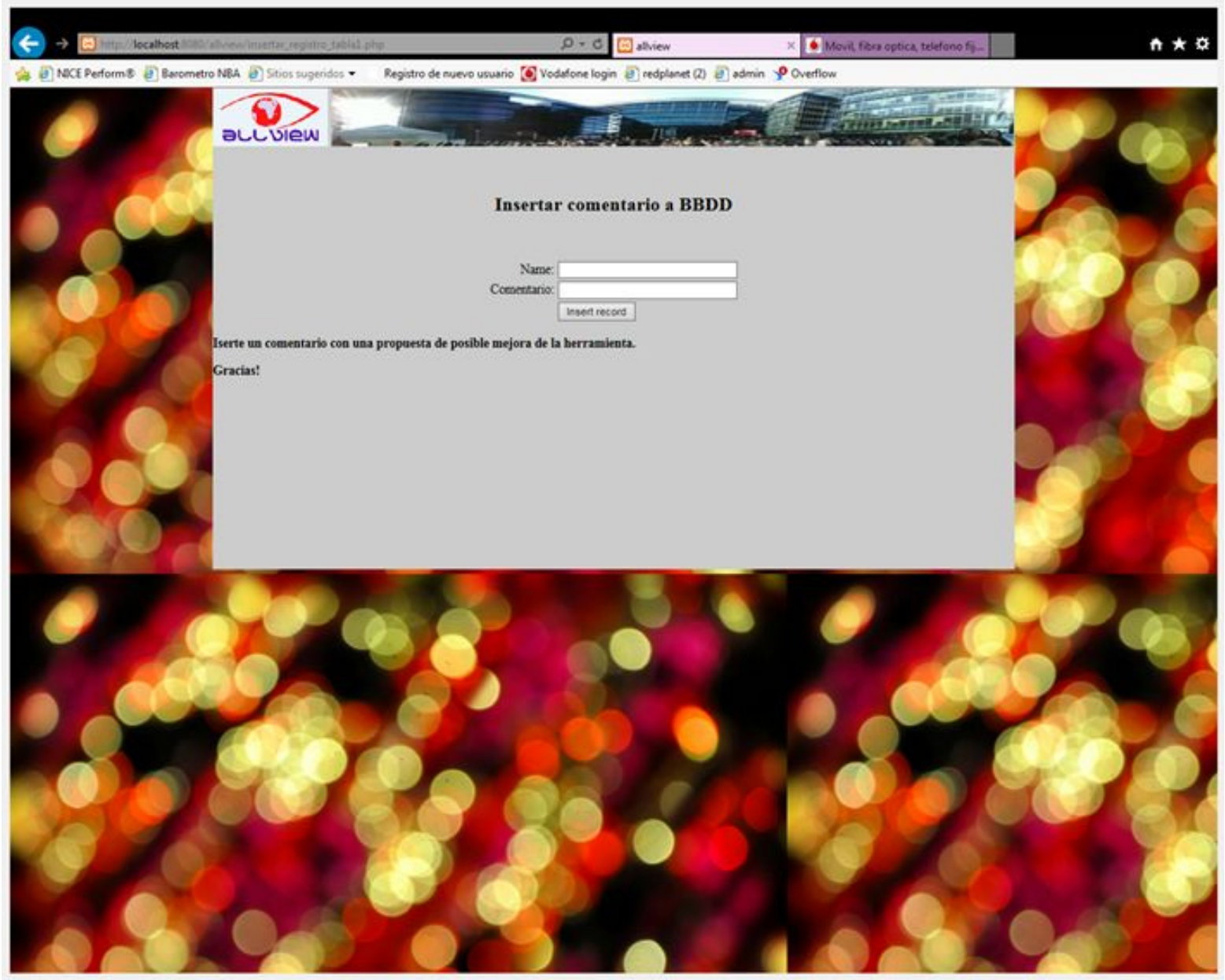

Ilustración 40: Interfaz web (Insertar comentario)

Ilustración 41: Interfaz web (Comentario insertado correctamente)

Para desarrollar esta opción he creado una tabla (*tabla1*) en nuestra BBDD con dos campos, nombre y comentario, especialmente para este fin.

Ilustración 42: Creación de tabla de Comentarios (*tabla1*)

Cuando el usuario, después de introducir la información, pincha en “Insert record”, la herramienta se conecta con esta tabla y deja los campos rellenados por el usuario en los registros correspondientes. El código que realiza esta operación está programado en *PHP* incluyendo consulta *SQL* que realiza el proceso de inserción de los registros en la tabla. Explicaré más en detalle cómo se realiza este paso y proporcionaré el código de este bloque en el capítulo de “Implementación”.

La siguiente opción que encontramos en el menú de navegación es “Equipo”. En esta opción, la idea es proporcionar la información general sobre el trabajo en equipo y en departamento. La información que colocaremos aquí e iremos actualizando sería relativa a los distintos proyectos que se realizan actualmente en el equipo y departamento y los logros de los últimos meses que se han alcanzado. La idea principal de esta información es por así decir marcar un poco el foco principal del trabajo que se está realizando actualmente, marcando los principales objetivos a corto y medio plazo.

Ilustración 43: Interfaz web (Equipo)

Las dos últimas opciones que quedan son enlaces a las páginas externas. Tal como están programadas, al pinchar en alguna de ellas nos abre una pestaña nueva del navegador con la

dirección web correspondiente a cada una de ellas. También se puede programar que la aplicación a la que está enlazada se abra en la misma pestaña o en una ventana de navegador nueva. Estas opciones son para facilitar al usuario los accesos a diferentes aplicaciones externas desde nuestra interfaz web.

Y finalmente la última de las opciones que nos queda, es también un enlace a una dirección web externa que contiene una herramienta utilizada por los usuarios de nuestra interfaz, permitiéndoles de este modo un acceso rápido y directo a ella para realizar las consultas posibles.

Con esto termino la descripción de la interfaz web. En el siguiente capítulo comentaré algunos de los aspectos que me parecieron más relevantes durante la realización del proyecto.

6.- Implementación

Como he mencionado antes, la idea de este capítulo es describir brevemente algunos aspectos que me parecieron más interesantes o más complicados durante el desarrollo de nuestra herramienta. Los aspectos que comentaré no estarán conectados entre sí, por lo que el orden en el que aparecerán no es obligatoriamente el mismo que el de proyecto.

Como he comentado en el capítulo anterior sobre la descripción de la interfaz web, en el apartado de "Consultas" disponemos de una opción para que los usuarios puedan insertar algún comentario acerca de la herramienta, proponiendo por ejemplo algún cambio o mejora. Esta opción es muy importante ya que nos permitirá mejorar la herramienta y ajustarla más a las necesidades de los usuarios.
La principal diferencia de este apartado con los demás apartados que interactúan con nuestra BBDD, como las consultas de gráficas para los usuarios, es que en este caso no vamos a leer nada desde la BBDD, sino al revés, vamos a insertar datos que nos proporcionará el usuario a una tabla. De este modo nosotros podemos ver y analizar estos comentarios.
El primer paso para implementar esta funcionalidad fue crear una tabla dentro de nuestra BBDD. Esta tabla es muy simple ya que solo contiene tres campos, el primero es un valor de identificador que es un campo autoincremental, y los otros dos son el nombre del autor del comentario, para poder saber a quién pertenece el comentario, y el propio comentario.
Una vez creada esta tabla, siguiente paso era la creación de un formulario dentro de nuestra interfaz web, para recoger los datos introducidos por el usuario. Con Dreamweaver podemos crear un formulario con las opciones disponibles en la pestaña "*Formulario*" o "*Forms*" en inglés.

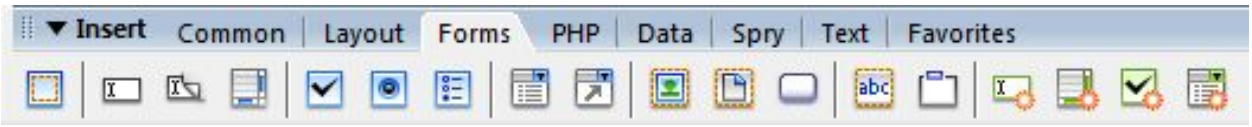

Ilustración 46: Opciones de Formularios (Dreamweaver)

En esta pestaña disponemos de múltiples opciones para crear un formulario como campos a rellenar por el usuario, opciones de marcar, listas desplegables, etc. En nuestro caso al ser un formulario muy simple he utilizado dos campos a rellenar por el usuario y un botón "*Insert record*" para subir la información introducida.

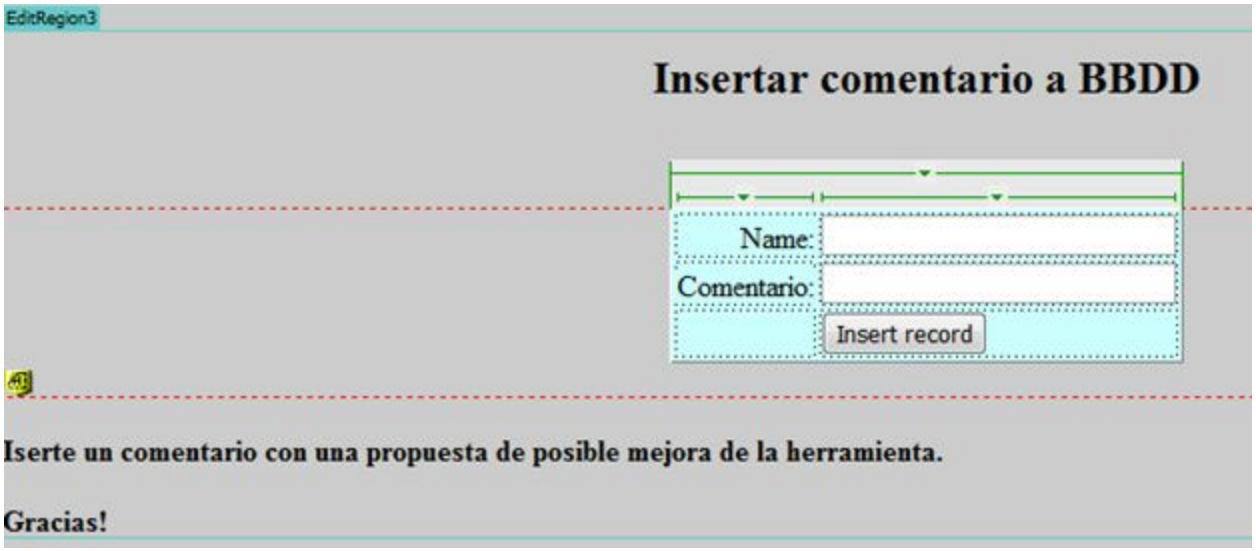

Ilustración 47: Creación del formulario (Diseño gráfico)

El código de este formulario queda como vemos en la siguiente imagen.

```
<h2 align="center">Insertar comentario a BBDD</h2>
<p> </p>

<form action="<?php echo $editFormAction; ?>" method="post" name="form1" id="form1">
  <table align="center">
    <tr valign="baseline">
      <td nowrap="nowrap" align="right">Name:</td>
      <td><input type="text" name="Name" value="" size="32" /></td>
    </tr>
    <tr valign="baseline">
      <td nowrap="nowrap" align="right">Comentario:</td>
      <td><input type="text" name="Comentarioo" value="" size="32" /></td>
    </tr>
    <tr valign="baseline">
      <td nowrap="nowrap" align="right"> </td>
      <td><input type="submit" value="Insert record" /></td>
    </tr>
  </table>
  <input type="hidden" name="MM_insert" value="form1" />
</form>
<p><strong>Iserte un comentario con una propuesta de posible mejora de la herramienta.</strong></p>
<p><strong>Gracias!</strong></p>
<!-- InstanceEndEditable --></div>
```

Ilustración 48: Creación del formulario (código)

Una vez tenemos esta parte diseñada, nos queda diseñar el mecanismo que se encarga de coger los datos introducidos por el usuario como variables. Este mecanismo comprueba si las variables realmente están pasadas, es decir definidas, y en caso afirmativo crea con ellas una sentencia en SQL para insertar las variables leídas en los campos correspondientes de la tabla de comentarios. Después de definir esta sentencia se establece la conexión con la BBDD y se lanza la consulta diseñada previamente con las variables a introducir.

```
32  $editFormAction = $_SERVER['PHP_SELF'];
33  if (isset($_SERVER['QUERY_STRING'])) {
34    $editFormAction .= "?" . htmlentities($_SERVER['QUERY_STRING']);
35  }
36
37  if ((isset($_POST["MM_insert"])) && ($_POST["MM_insert"] == "form1")) {
38    $insertSQL = sprintf("INSERT INTO tabla1 (Name, Comentario) VALUES (%s, %s)",
39                         GetSQLValueString($_POST['Name'], "text"),
40                         GetSQLValueString($_POST['Comentario'], "text"));
41
42    mysql_select_db($database_mi_conexion_local, $mi_conexion_local);
43    $Result1 = mysql_query($insertSQL, $mi_conexion_local) or die(mysql_error());
44
45    $insertGoTo = "insercion_registro_tabla1_ok.html";
46    if (isset($_SERVER['QUERY_STRING'])) {
47      $insertGoTo .= (strpos($insertGoTo, '?')) ? "&" : "?";
48      $insertGoTo .= $_SERVER['QUERY_STRING'];
49    }
50    header(sprintf("Location: %s", $insertGoTo));
51  }
```

Ilustración 49: Creación de formulario (Generación y envío de la sentencia SQL)

Vemos que después de lanzar la sentencia, indicamos en la línea 45 a que página nos tiene que llevar tras inserción de los datos.

Durante el Desarrollo de la interfaz, sobre todo al principio, cuando todavía no estamos familiarizados con el código *PHP*, tenemos muchos *"error"*, *"warning*" y *"notice"* que nos aparecen en la interfaz tras ejecutar el código en el momento de carga de la página. Si se trata de errores, los tenemos que resolver para continuar con el desarrollo. Los demás los podemos evitar reconfigurando que no nos aparezcan dichos avisos. Esta configuración se realiza en el fichero de nuestro servidor Apache y se llama "php.ini". Sin hacer esta configuración nos encontraríamos con la siguiente situación en nuestra interfaz:

Ilustración 50: Interfaz web (Warning, Notice)

Para solucionarla, en el fichero “php.ini” modificamos siguiente línea de código añadiendo campos nuevos.

```
; Common Values:
;  E_ALL (Show all errors, warnings and notices including coding standards.)
;  E_ALL & ~E_NOTICE  (Show all errors, except for notices)
;  E_ALL & ~E_NOTICE & ~E_STRICT  (Show all errors, except for notices and coding standards warnings.)
;  E_COMPILE_ERROR|E_RECOVERABLE_ERROR|E_ERROR|E_CORE_ERROR  (Show only errors)
; Default Value: E_ALL & ~E_NOTICE & ~E_STRICT & ~E_DEPRECATED
; Development Value: E_ALL
; Production Value: E_ALL & ~E_DEPRECATED & ~E_STRICT
; http://php.net/error-reporting
error_reporting=E_ALL & ~E_DEPRECATED & ~E_STRICT &~E_NOTICE &~E_WARNING
```

Ilustración 51: Interfaz web. Modificación de fichero "php.ini"

Después de realizar este cambio y para que empiece a funcionar tenemos que parar nuestro servidor Apache y arrancarlo de nuevo ya que es leído al arrancar el servidor. Una vez hecho esto, la página aparecerá sin avisos de *“warning”* y *“notice”*.

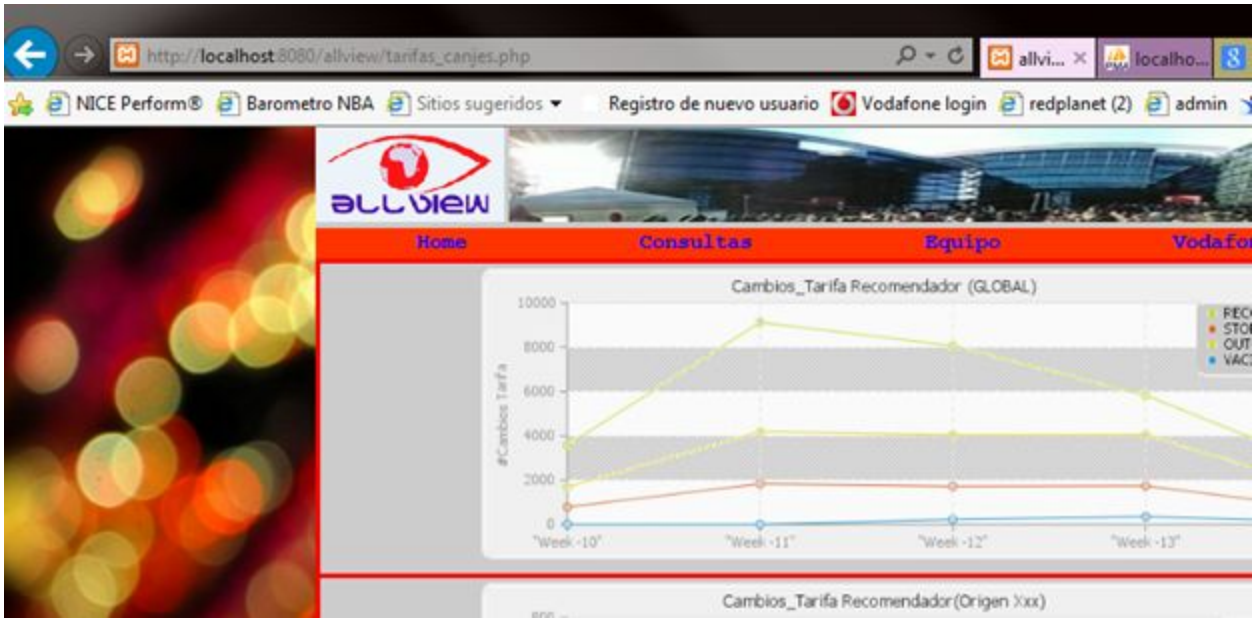

Ilustración 52: Interfaz web (Sin Warning y Notice)

Además de esta configuración se puede configurar muchos más parámetros con el fichero php.ini como por ejemplo el número máximo de memoria que un script puede consumir, configuración de un buffer intermedio, etc. Pero por la específica del proyecto realizado estas configuraciones no son necesarias, por lo que quedan fuera de esta memoria.

Para realizar el proyecto, en vez de instalar MySQL y Apache por separado, podemos utilizar una plataforma de servidor que ya contiene integrados en sí estas dos herramientas y varias más, aunque estas otras no las utilizaremos. Utilizando esta opción, de una plataforma del servidor, evitamos posibles incompatibilidades de versiones de las herramientas que vamos a utilizar, ya que las versiones que vienen incluidas en ella son ya compatibles. Otra facilidad que nos da es tener el control de todas ellas en un sitio. En nuestro caso, entre varias opciones de las que disponemos hemos elegido una plataforma llamada XAMPP ya que cumplía con nuestras necesidades.

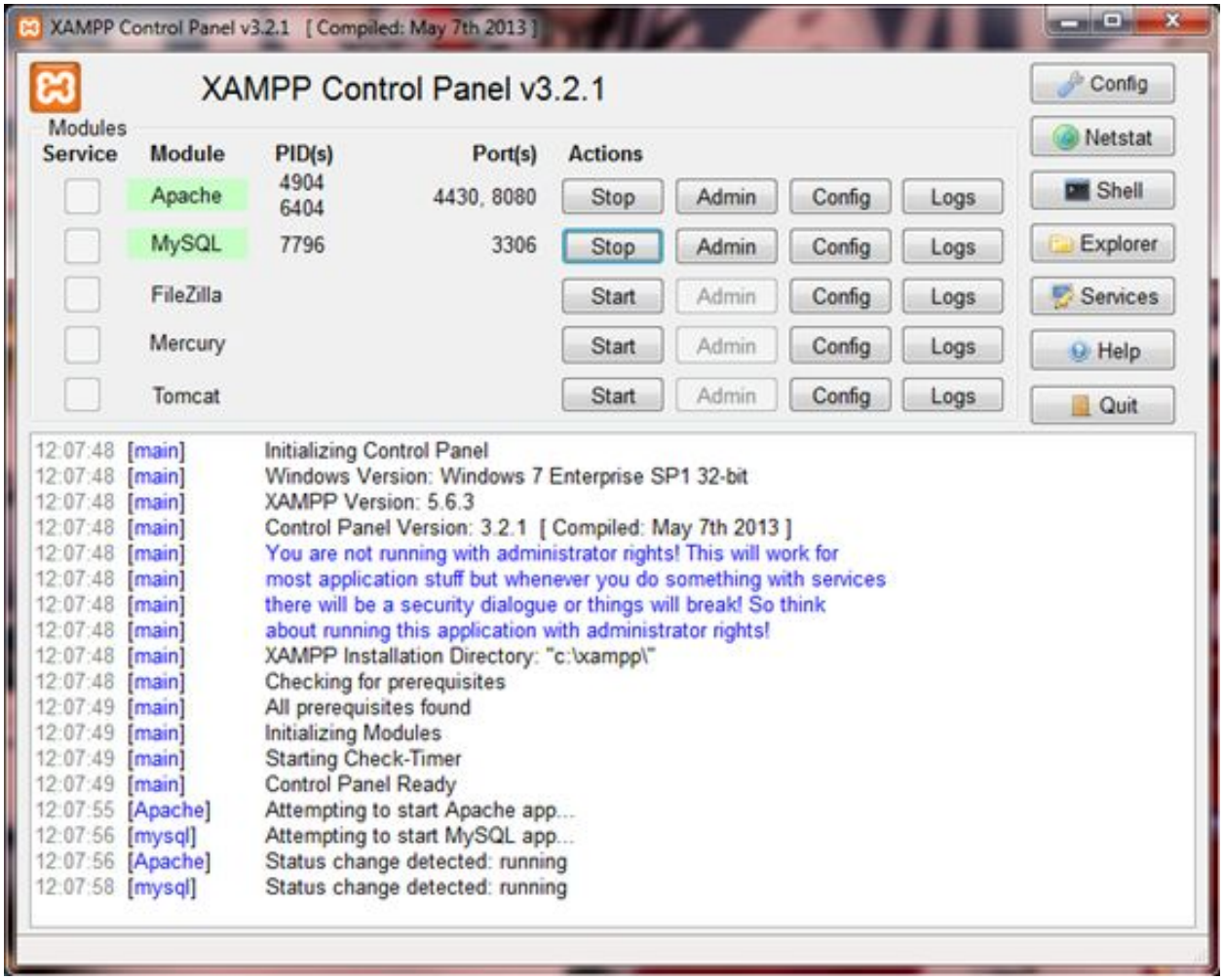

Ilustración 53: Plataforma de servidor XAMPP

Ahora bien, una vez que tenemos nuestras herramientas instaladas con XAMPP o por separado, al arrancarlas nos puede surgir un error debido a que los puertos preconfigurados que utilizan estas herramientas están ocupados por otras de nuestro ordenador. Para resolver este problema tenemos que reasignar los puertos utilizados por nuestras herramientas. En nuestro caso tenía que reasignar los puertos preconfigurados del servidor Apache, que son 80 y 443. En mi caso he cambiado el puerto 80 por 8080 y el 443 por 4430. Para modificar estos puertos tenemos que cambiarlos algunas líneas en los ficheros httpd.conf y httpd-ssl.conf.

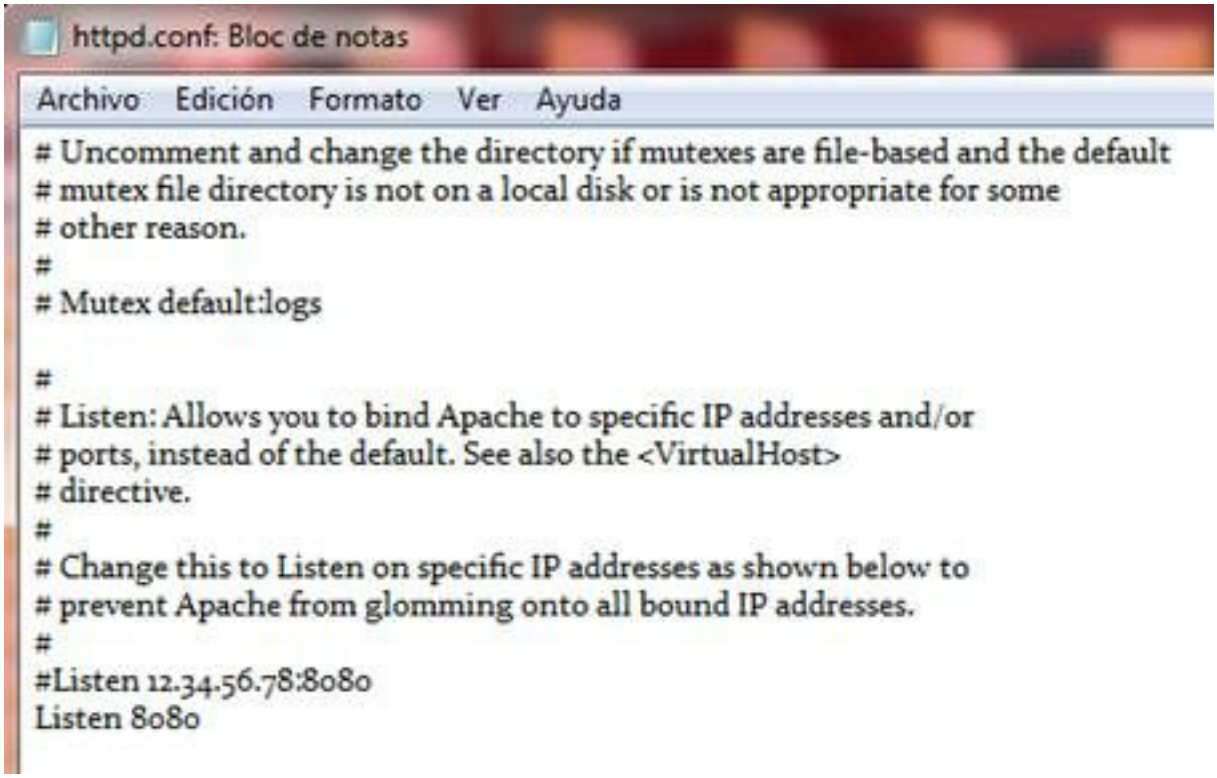

Ilustración 54: Configuración puertos Apache 1 (httpd.conf)

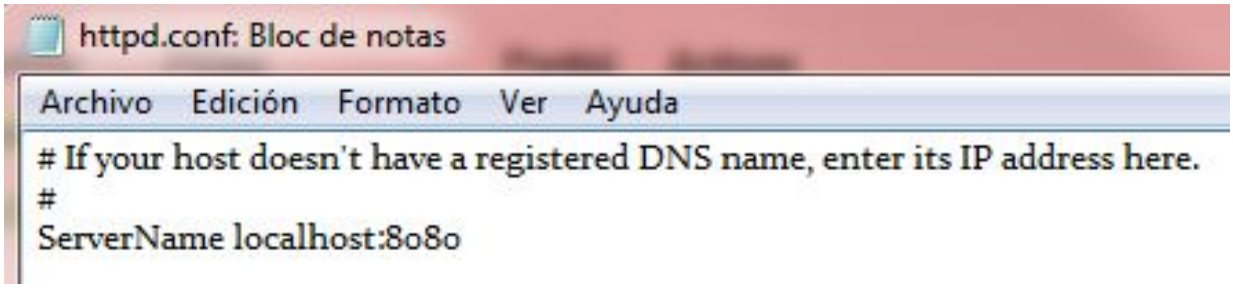

Ilustración 55: Configuración puertos Apache 2 (httpd.conf)

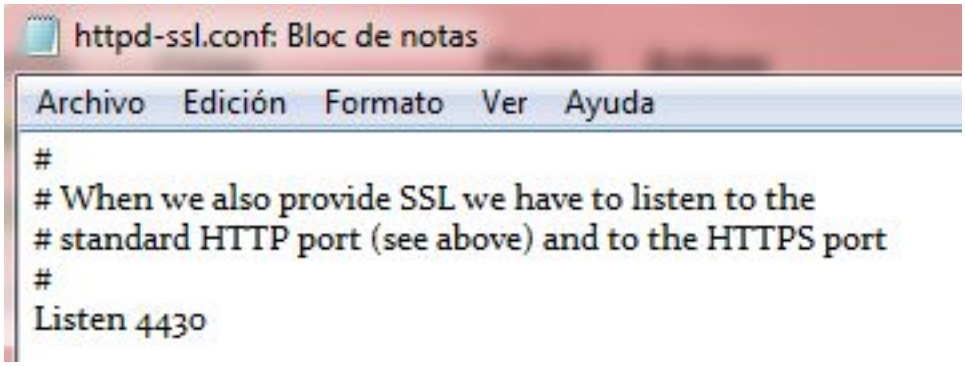

Ilustración 56: Configuración puertos Apache 3 (httpd-ssl.conf)

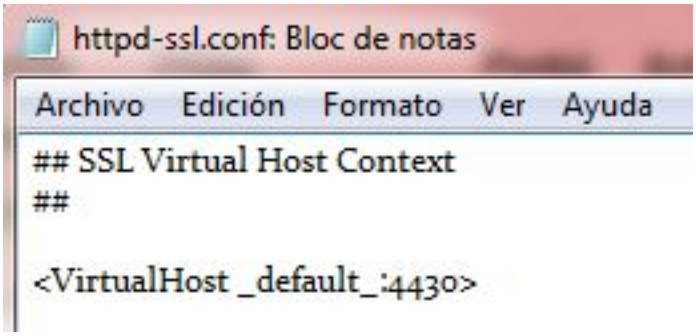

Ilustración 57: Configuración puertos Apache 4 (httpd-ssl.conf)

Una vez hechos estos cambios arrancamos nuestro servidor de nuevo y el problema tiene que quedar resuelto.

Como he hablado ya en varias ocasiones durante la memoria del proyecto, es mucho más cómodo sacar los datos ya preprocesados cuando los solicitamos estando en el lado de la interfaz web. Esto era porque el procesado en el lado de la interfaz, es decir en php, es más lento y más difícil de realizar. Pero claro, con mucho que simplificamos los datos en nuestra BBDD en algunos momentos para consultas específicas las tendremos que procesar más en PHP antes de pasarlos al bloque de generación de gráficas.
Uno de estos procesados he tenido que hacer para la representación de los datos de un piloto (una campaña de prueba con un grupo de agentes durante un periodo de tiempo) junto con los datos normales y corrientes que no tienen carácter periódico.
La problemática fue que el piloto solo se realizaba durante varias semanas y para representar los datos del mismo con los demás no podíamos pasar estos datos como una serie de datos simple ya que su longitud no era la misma que la de demás datos. Esto era debido a que en los periodos que el piloto no estaba activo, es decir no tenía ninguna actividad, al no tener ningunos valores correspondientes a estas fechas las consultas SQL no nos devolvian ceros para estos fechas, sino no nos devolvian nada. Por lo que al pasarlos de esta forma se hubiesen colocado al principio del eje de semanas y los valores no hubiesen correspondido con semanas a los que corresponden. Es decir, imaginamos que representamos datos del piloto con los datos normales en un periodo de cuatro semanas, pero el piloto solo se realizaba en semanas dos y tres. La consulta nos devuelve dos valores correspondientes a la semana dos y tres. Si no hacemos nada y pasamos los datos del piloto con los demás, el bloque de generación colocará estos datos de las semanas como si fuesen correspondientes a la semana uno y dos.
Para resolver esta problemática la idea era rellenar las semanas en las que el piloto no estaba activo con ceros, y las demás semanas dejar con los valores que tenían. Para realizar esto teníamos que comparar las semanas que teníamos de nuestro eje de gráfica con las semanas que devolvía consulta de piloto que suponía que tenía alguna actividad durante estas semanas. Si estas semanas coincidían dejábamos para esta semana el valor que tenía el piloto y lo pasábamos a un array nuevo de piloto en la posición correspondiente de semana. En el caso

en que la semana del eje no estaba en el eje del piloto significaba que el piloto no era activo en esta semana, por lo que se pasaba un cero al array nuevo de piloto para esta semana. Una vez hecha esta comprobación y generado el array nuevo de piloto con ceros en las semanas en las que no estaba activo, pasábamos este array con los demás al bloque de generación de las gráficas. A continuación muestro el fragmento del código *php* correspondiente a este proceso.

```
//Procesado para rellanar con 0 las semanas del piloto en las que no estaba activo
//print("Foreach3");
$done=false;
$vaciosreal=array(); //nuevo array de piloto
foreach($ejesemana as $i){ //semanas reales
 foreach($vacios_week as $w){ //semanas del piloto
 if($i == $w){
 $pos = array_search($w, $vacios_week);
  array_push($vaciosreal, $vacios[$pos]);//valor del piloto
  $done=true;
 }}
 if($done==false) array_push($vaciosreal, 0);//0
  $done=false;
 }
```

Ilustración 58: Fragmento de procesado en php (datos de piloto)

Con esto termino todo lo que quería comentar en el capítulo de “*Implementación*”.

7.- Validación / pruebas

En este capítulo describiré qué pruebas de validación de la herramienta he realizado a lo largo del desarrollo del proyecto. Por la específica del proyecto realizado no teníamos que diseñar algunas baterías específicas de pruebas como las que podrían ser utilizadas en proyectos estadísticos, de análisis o por ejemplo de reconstrucción de imágenes. En nuestro caso las pruebas realizadas consistían básicamente en verificación de que cada una de las partes de la herramienta funciona tal como era planificado que funcionase.

- Empezando por el *Bloque1*, la principal validación que teníamos que realizar es comprobar que, una vez diseñada nuestra BBDD con las tablas contenidas en ella, funcionan los procesos de carga y actualización de datos. Como he comentado antes, como una opción para realizar esta tarea fue diseñado un script en código *Python* para automatizar este proceso por si la persona que lo va realizar no tuviese conocimientos de área de SQL. De este modo teníamos que comprobar que el proceso de carga de datos se realiza bien tanto vía script como utilizando otras herramientas como phpMyAdmin o MySQL Query Browser. Por ello, con cada una de estas opciones hemos realizado varias operaciones de carga del fichero comprobando que efectivamente el proceso se realiza sin ningún tipo de error. Para comprobar que esto efectivamente es así, después de cada proceso de carga comprobamos el contenido de nuestra tabla verificando que los datos están cargados o actualizados mediante una consulta hacia la misma.
- El siguiente aspecto importante de la validación es comprobar la conectividad entre nuestra interfaz web y la BBDD una vez que está configurada. Para verificar que la conectividad realmente está establecida nos basta con comprobar que se está realizando el proceso de lectura de datos desde la BBDD una vez hemos lanzado la consulta hacia ella o verificar que los cambios que estamos haciendo en nuestra BBDD desde la interfaz web realmente se quedan reflejados en la BBDD con carácter permanente. Para hacer este tipo de verificación he utilizado la segunda opción, comprobar que los cambios que efectuamos desde la interfaz se quedan reflejados en la BBDD. He creado una tabla simple en nuestra BBDD y desde la interfaz web he agregado datos hacia ella, posteriormente a través de phpMyAdmin o MySQL Query Browser podemos ver que efectivamente los datos introducidos están dentro de la tabla, lo que significa que la conexión fue realizada con éxito.
- Aunque la comprobación anterior es más profunda y por así decir muestra más interconectividad entre la interfaz y la BBDD, existe una manera más simple. Podemos comprobar si la conexión está establecida directamente con Dreamweaver, ya que proporciona una opción para esto en la propia configuración de la conexión. Una vez

introducidos los parámetros de la conexión, en el menú tenemos una opción que se llama “Test”. Al pinchar en ella Dreamweaver comprobará si la conexión se establece y en el caso afirmativo nos mostrará siguiente el mensaje.

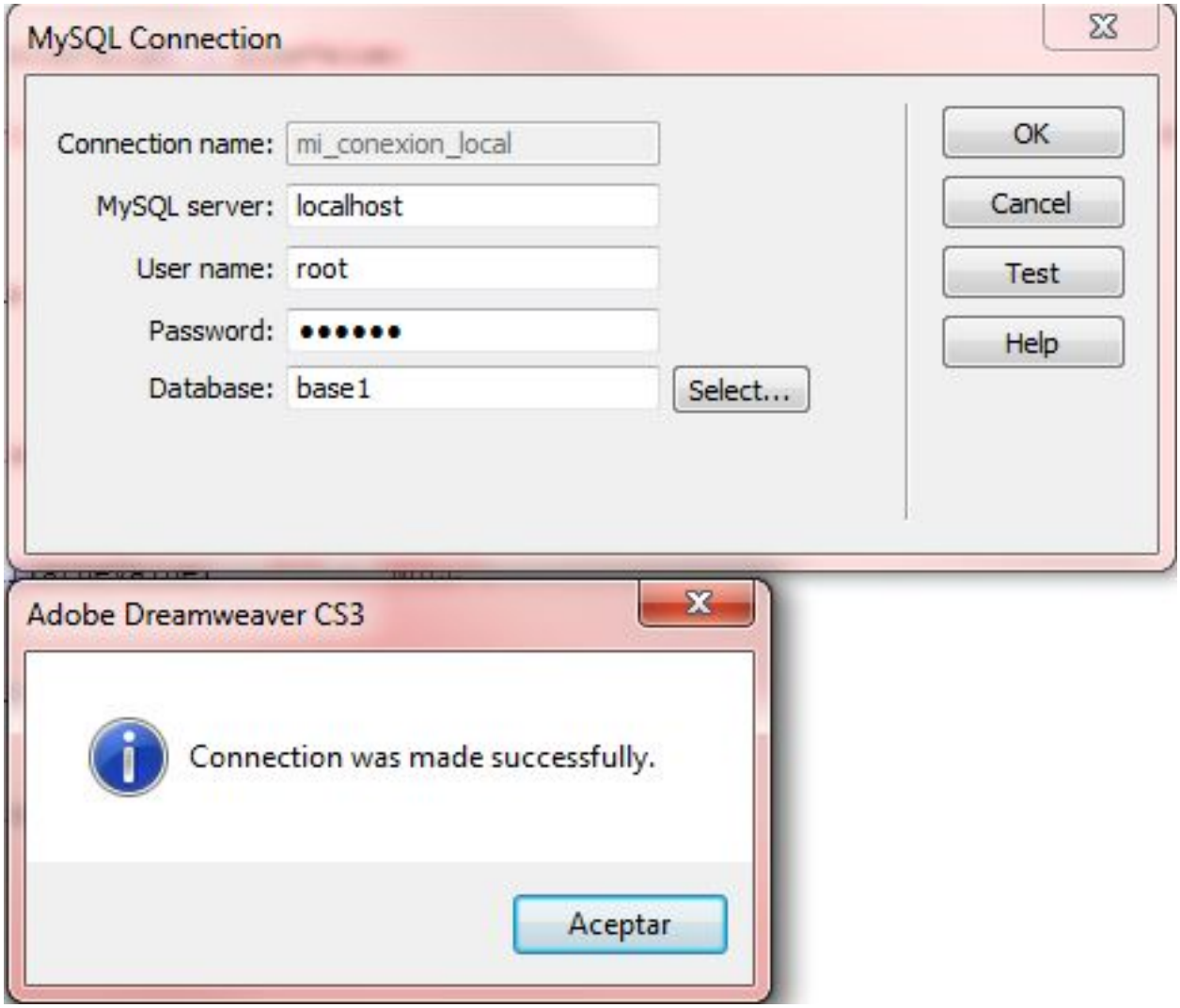

Ilustración 59: Comprobación de la conexión entre la Interfaz web y la BBDD (Dreamweaver)

- En la fase de desarrollo de las consultas era muy importante comprobar que las consultas diseñadas recogen toda la información requerida e indicada en las mismas, por lo que era muy importante comprobarlas antes fuera del entorno de la interfaz y solo después integrarlas con el código *PHP*. Este tipo de comprobaciones se realizaba básicamente con phpMyAdmin y con MySQL Query Browser. Una vez comprobado que las consultas tienen sintaxis correcta y devuelven la información que necesitamos en cada momento, procedemos a integrarlas dentro del código *PHP*.
- Durante la generación de las gráficas, la principal verificación que teníamos que comprobar es que las gráficas muestran los datos que realmente pasamos al bloque de generación de las mismas. Es decir, teníamos que comprobar que los datos que

pasábamos al bloque de generación estaban bien procesados y bien entendidos y no había ningún problema como por ejemplo incompatibilidad de formato, etc. Esta verificación era muy simple ya que solo teníamos que verificar si los datos mostrados en las gráficas corresponden a los datos que devuelven las consultas.

- La última y la más importante verificación es la que incluye la comprobación de todos los pasos anteriores. Esta verificación incluye actualización de los datos en la BBDD y visualización de dicha actualización en las gráficas que vería el usuario. Consiste en actualizar la información de algunos datos que estamos monitorizando en la gráfica y ver que tras esta actualización nuestra gráfica también se actualiza al cargarla de nuevo. Este tipo de comprobación permite comprobar toda la cadena de la funcionalidad del sistema, empezando por la actualización de datos, pasando por la interconexión de los bloques y finalizando con el bloque de generación de las imágenes y la actualización de las mismas.

8.- Conclusiones y trabajos futuros

8.1.- Conclusiones (½ o 1 página)

- *Que se ha cumplido con los objetivos iniciales*
- *Dificultades*

La conclusión principal del trabajo que se ha realizado es que al final todos los objetivos principales que teníamos al principio del proyecto los hemos podido cumplir. La herramienta desarrollada resuelve las necesidades para las que estaba planteada, que básicamente son el manejo de grandes volúmenes de datos, procesamiento de los mismos y muestra de los resultados en forma de gráficas mediante la interfaz web desarrollada.
Durante el proyecto han ido surgiendo diferentes dificultades prácticamente en todas las fases realizadas. Las dificultades en su mayor parte venían de que las tecnologías utilizadas en el proyecto eran en su mayor medida o nuevas para mí o solo tenía conocimientos teóricos sobre ellas, por lo que durante de la realización del proyecto había que ir probando y aprendiendo muchas cosas nuevas, ver cómo se configuran, cómo funcionan y cómo interaccionan entre sí.
Con total seguridad puedo decir que estoy muy contento con todo lo que he hecho y todo que he aprendido, ya que esto me da nuevas posibilidades en el trabajo y estoy seguro de que las utilizaré en el futuro. De hecho, algunos de los conocimientos adquiridos durante el proyecto, como el diseño y la administración de las BBDD, ya los he utilizado en mi trabajo en ocasiones puntuales.

8.2.- Trabajos futuros

Realmente siendo el tipo de proyecto que hemos realizado existen muchas posibles mejoras que podemos realizar. Las podemos dividir en varios tipos: mejoras del rendimiento, mejoras de la funcionalidad o del uso que el usuario puede dar a la herramienta y mejoras de la percepción de la herramienta por parte del usuario.
Mencionaré varias posibles mejores de cada tipo que podemos realizar.

- Mejoras de rendimiento:

    1. Mejoras de rendimiento que permiten reducir el tiempo de procesamiento de datos y el tiempo de su representación. De este modo conseguiremos que nuestra herramienta funcione más rápido y sea más eficiente.
       Uno de los puntos que podemos mejorar para conseguir este objetivo es la parte de código correspondiente a la obtención de los datos y a su procesamiento. En este paso tenemos una parte de código SQL correspondiente a la consulta que lanzamos a la BBDD para obtener los datos. Este código SQL está integrado dentro del código PHP. El problema está en que para representar algunos gráficos necesitamos obtener datos correspondientes a diferentes series a representar (ej.: agencias, tarifas, etc). Es decir, si nuestra gráfica representa el rendimiento de cinco tarifas diferentes, necesitamos constituir cinco consultas SQL y lanzarlas hacia la BBDD una a una estableciendo la conexión con la base de datos en cada iteración. Este modelo de obtención de los datos fue diseñado así por dos motivos. El primer motivo consiste en que el bloque de representación de las gráficas necesita que le pasamos por separado los datos de cada una de las series, es decir, pasar cinco series de datos una a una sin tener la posibilidad de pasar el bloque entero con todos los datos. El segundo motivo fue la necesidad de reducir el procesamiento de los datos obtenidos en el lado de la interfaz. Es decir, intentar conseguir que los datos que devuelve la consulta coincidan al máximo con los que necesitamos para la serie correspondiente, evitando así el procesamiento de los mismos. Una primera solución fue la obtención de todos los datos que necesitamos con una única consulta, y posterior separación de estos datos en las series correspondientes en el lado de la interfaz. Esta primera solución tenía el problema de tardar un tiempo notable en la ejecución. Por este motivo hemos llegado a la solución que tenemos actualmente. La mejora en este punto podría ser conseguir un bloque de representación de las gráficas que daría más flexibilidad a la hora de pasarle datos a representar. Esto ahorraría el tiempo de procesamiento en el lado de la interfaz y reduciría el número de consultas necesarias por una gráfica. Otra de las posibles opciones sería utilizar un framework de Django para el procesamiento de datos y la generación de los gráficos.

    2. Otro punto de mejora que nos permitirá reducir el tiempo de procesamiento, y dependiendo de la información que queremos visualizar nos permitirá evitar la

necesidad de realizar una consulta a la BBDD, es introducción de una caché para el almacenamiento temporal de los datos. Lo explicaré con dos ejemplos sencillos. Supongamos que en el primer ejemplo hemos consultado un gráfico con los valores hasta el mes en curso, es decir, nuestra aplicación ha lanzado la consulta hacia nuestra BBDD, ha procesado los datos devueltos y nos ha devuelto este gráfico. Ahora imaginemos que al día siguiente entramos de nuevo a la herramienta para consultar el mismo gráfico. Entonces, la aplicación vuelve a repetir los mismos pasos de antes para mostrar el mismo gráfico. Podríamos evitar repetir todos estos pasos si después de la primera ejecución almacenamos los datos en una caché, y cuando necesitamos de nuevo mostrar el gráfico reutilizamos estos datos. En el segundo ejemplo sería todo igual que en el primero, pero suponiendo que cuando solicitamos el gráfico tenemos datos del nuevo mes que todavía no hemos analizado. En este caso podríamos reutilizar los datos de los meses anteriores y solo consultar los datos del nuevo mes. En este punto cabe destacar que cuando llevaremos unos pocos meses y el volumen de los datos no sea relativamente alto, probablemente no notaremos el efecto de carga y de procesamiento de datos. Pero mirando a largo plazo tenemos que tener en cuenta que tendremos un histórico de datos de volumen notable que cada vez irá creciendo más, por lo que el proceso de extracción de los datos y su procesamiento serán cada vez más notables. Tener una caché nos permitirá hacer este proceso más rápido y más eficiente. Además quiero recordar que tal como está diseñada esta parte ahora, después de hacer la extracción de los datos y mostrar el gráfico, este gráfico se guarda en el directorio local y la interfaz web lo lee desde una carpeta local, por lo que si estamos cambiando de ventanas dentro de la interfaz web y luego otra vez volvemos a la ventana del primer gráfico, la herramienta no lo vuelve a recalcular, ya que lo coge desde la carpeta local. Si queremos refrescar el gráfico, es decir, volver a lanzar la consulta de datos, tenemos la opción de refrescar los datos con el botón de “reload”.

3. Un punto más sobre el que se puede trabajar y sobre el que se trabaja actualmente es optimización de los datos de los ficheros fuente que nos llegan. Estos ficheros fuente se generan por el personal de otro departamento y se distribuyen a muchos departamentos y equipos, incluido el nuestro. Realmente no es fácil introducir algún cambio en el formato de datos de estos ficheros, ya que unos campos de estos pueden presentar un gran interés y ser analizados por integrantes de un equipo y al mismo tiempo no tenerse en cuenta por los integrantes de otros equipos. En lo que se está trabajando actualmente es en intentar llegar a un formato de los ficheros que sea cómodo de manejar para todos, intentando disminuir diferentes cruces con otras bases de datos y tablas para obtener el formato con campos necesarios. Durante el desarrollo del proyecto el formato de la BBDD ha sufrido varios cambios y se prevé que sufrirá más cambios en el futuro.

- Mejoras de funcionalidad:
  Aquí sobre todo hablamos sobre aumentar la funcionalidad de la herramienta, haciéndola más flexible en cuanto a los resultados que muestra y quizás integrándola mejor con otras herramientas que se están utilizando en el departamento.

    1. Más flexibilidad en la visualización de las gráficas cambiando variables a visualizar por el usuario. Dar más funcionalidad para el usuario significa permitirle visualizar cuantas cosas le pueden interesar. Teniendo en cuenta que estas necesidades varían según el usuario, la idea para mejorar sería permitir introducir parámetros variables que él usuario puede elegir para seleccionar el contenido de las gráficas. Esto significa introducir elementos de interacción entre el usuario y la interfaz para que el usuario pueda parametrizar la consulta introduciendo parámetros de diferentes campos. Este elemento permitiría reducir el número de las gráficas estáticas, es decir las gráficas que están diseñadas por nosotros. El desarrollo de esta parte supondrá introducir cambios en el diseño de la interfaz web introduciendo una serie de listas desplegables y campos a seleccionar por el usuario para elegir los parámetros de las gráficas. Por otro lado supondrá desarrollo del código php (u otro código como por ejemplo python) que permitirá introducir estos parámetros preelegidos por el usuario en las consultas de SQL que posteriormente se enviarán a la BBDD. Una vez recibidos los datos seleccionados probablemente necesitaremos desarrollos adicionales para el bloque de procesamiento y de representación de las gráficas. Este diseño no fue introducido desde el principio ya que suponía más tiempo de desarrollo y queríamos sacar una primera versión de la herramienta para poder visualizar los parámetros básicos que nos interesan. Además, cuanto antes dispongan de la herramienta, antes los usuarios la empezarán a utilizar y nos irán dando realimentación sobre las mejoras posibles. Para introducir este cambio es necesario realizar una serie de reuniones con los potenciales usuarios de la herramienta para recoger los atributos o parámetros que ellos deseen incluir en la misma. Este proceso requiere tiempo, y como no es fundamental lo hemos decidido dejar para más adelante, cuando tengamos la primera realimentación de los usuarios y cuando empezaremos introducir primeras modificaciones solicitadas.

    2. Otra de las cosas que aumentaría la funcionalidad de nuestra herramienta es la introducción de un historial de consultas detalladas por mes o por el cuatrimestre para poder analizar con más detalle el rendimiento de unos u otros KPIs en los periodos anteriores. Esta opción podrá ahorrar el tiempo de usuario, ya que no tendrá que solicitar el cálculo de estas gráficas. Por otro lado, aumentará el rendimiento de la herramienta, ya que evitaremos realizar consultas innecesarias a la BBDD y perder el tiempo en el procesamiento y la representación de las gráficas. Como opción estas gráficas podrían calcularse (automáticamente o

manualmente) una vez finalizado el mes o el cuatrimestre y almacenarse en el directorio local para que la interfaz les pudiese coger desde allí.

3. Teniendo en cuenta la multitud de los indicadores que estamos analizando y que unos de ellos influyen directamente en los otros, a veces para explicar el rendimiento de un KPI tenemos que consultar el de otros KPIs que no siempre son manejados por nuestro equipo. Por ello a menudo tenemos que solicitar esta información a otros equipos o consultarla en otras herramientas diseñadas por estos equipos que nos pueden facilitar la información solicitada. Para facilitar este tipo de acciones y hacer nuestra herramienta más completa, es decir, reducir el tiempo para la obtención de datos y evitar solicitudes innecesarias a otros equipos, estamos analizando las posibilidades de integrar nuestra herramienta con otras que ya se están usando. Realmente esta integración consiste en la integración de nuestra herramienta con las BBDD de otros equipos donde está almacenada la información de interés. Actualmente estamos analizando la posibilidad de integrar nuestra herramienta con la BBDD de otro equipo. Es la BBDD que da la información a otra herramienta interna de otro equipo que a menudo tenemos que usar. Si finalmente conseguiremos esta integración podríamos tener los datos que nos interesan dentro de nuestra herramienta y en el formato y rangos que nos interesan. De este modo el análisis sería más fácil, ya que tendríamos toda la información en el mismo sitio y al mismo tiempo. Esta integración supondría unos cambios en el diseño de la interfaz web para introducir nuevas posibilidades, y por debajo de esto supondría el diseño de nuevas consultas y de la conexión con la nueva BBDD. Al pertenecer estas BBDD a otros equipos y departamentos, para llevar a cabo esta acción ante de todo tenemos que tener permisos para realizar conexiones a estas BBDD, lo que a veces no es fácil de conseguir. En el caso de que tengamos este tipo de permisos, el administrador de las BBDD de interés tendría que habilitar para nosotros un usuario dándole ciertos permisos para el manejo de las BBDD.

- Mejoras de la percepción de la herramienta por parte del usuario:
  Aquí se trata básicamente de la parte con la que interacciona el usuario, es decir, la interfaz web. Las mejoras tienen que ver con el diseño de este bloque.

  1. Personalizar el diseño de la interfaz desarrollando elementos gráficos propios. Es una cuestión más de estética que de otra cosa. Para llevar a cabo esta tarea se puede utilizar una multitud de herramientas que permitan diseño de elementos gráficos web. En nuestro caso utilizaremos el Adobe Fireworks CS4, ya que es de la misma familia que herramienta Dreamweaver utilizada para el desarrollo de la interfaz web. Una vez diseñados estos elementos personalizados los insertaremos en el diseño de nuestra interfaz. En este punto

para trabajar sobre el diseño de la interfaz tendremos que utilizar las “Hojas de estilos” de Dreamweaver que se llaman CSS.

2. Un punto más que mejora la percepción del usuario es un diseño más cómodo y más simple, para conseguir que la página sea muy intuitiva para el usuario. Tenemos que evitar demasiadas pestañas y hojas dentro de nuestra interfaz, para evitar que el usuario se pierda pasando de una a otra. Además, con el desarrollo y nuevas funcionalidades que iremos dando a la herramienta, tenemos que diseñar la interfaz de tal modo que gráficas y datos que están incorreladas entre sí pero se muestran en el mismo lugar, es decir, dentro de la misma hoja para no tener que cambiar de una a otra. Esto permitirá a los usuarios analizar los datos de una forma más cómoda y más rápida.

9. Presupuesto

Para realizar el presupuesto de nuestro proyecto he utilizada la plantilla disponible en la web oficial de la Universidad Carlos III de Madrid. Esta plantilla incluye todos aspectos que tenemos que incluir en el presupuesto como la duración del proyecto, coste personal, coste de equipos y licencias de softwares y amortiguación de los mismos, costes indirectos del proyecto e IVA.

Como hemos podido ver en la planificación del proyecto, la longitud del proyecto es de unos 4 meses y media durante los cuales en algunos momentos del mismo yo tenía que reunirse con mis compañeros para hablar sobre diferentes aspectos del mismo. Este tiempo dedicado a las reuniones y a la resolución de las dudas sobre el desarrollo de proyecto he contado como medio mes de dedicación al mismo.

Cabe también destacar que muchos de los softwares utilizados a lo largo del proyecto son softwares libres, por lo que no suponen ningún coste al proyecto.

En seguida muestro una tabla detallada del presupuesto.

UNIVERSIDAD CARLOS III DE MADRID
Escuela Politécnica Superior

PRESUPUESTO DE PROYECTO

1.- Autor:		Artem Motsarenko				
2.- Departamento:		Ingeniería Telemática				
3.- Descripción del Proyecto:						
- Título	Procesamiento y muestra de los KPIs de Performance mediante una Base de Datos y una Herramienta Web					
- Duración (meses)		4,5				
Tasa de costes Indirectos:		20%				
4.- Presupuesto total del Proyecto (valores en Euros):						
	Euros					
5.- Desglose presupuestario (costes directos)						
PERSONAL						
Apellidos y nombre	N.I.F. (no rellenar - solo a título informativo)	Categoría	Dedicación (hombres mes) [a]	Coste hombre mes	Coste (Euro)	Firma de conformidad
					0,00	
Myriam Xxx		Ingeniero Senior	0,5	4.289,54	2.144,77	
Artem Motsarenko		Ingeniero	4,5	2.694,39	12.124,76	
Oscar Xxx		Ingeniero (soporte)	0,5	2.649,39	1.324,70	
					0,00	
		Hombres mes	5,5	Total	15.594,22	
			825	horas		
[a] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)						
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)						

Ilustración 60: Presupuesto 1

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Ordenar Servidor para el desarrollo	500,00	50	5	60	20,83
Resto de Hardware(pantalla, raton)	150,00	50	5	60	6,25
Licensia Windows 7	120,00	50	5	60	5,00
Licensia paquete Office	109,00	10	5	60	0,91
Licensia Dreamweaver CS3	555,64	100	5	60	46,30
					0,00
	1434,64			Total	79,30

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO[e]

Descripción	Empresa	Costes imputable
	Total	0,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas,

6.- Resumen de costes

Presupuesto Costes Totales	Presupuesto Costes Totales
Personal	15.594
Amortización	79
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	3.135
Total	18.808
IVA (21%)	3.950

Ilustración 61: Presupuesto 2

Como vemos el presupuesto final del proyecto nos queda en *18.808€* más *3.950€* de IVA.

Lista de acrónimos

COPS: Comercial Operations (Operaciones Comerciales)
KPI: Key Performance Indicator (Indicador Clave de Rendimiento)
NBA: Next Best Action (Siguiente Mejor Acción)
Msisdn: Mobile Station Integrated Service Digital Network (Estación móvil de la Red Digital de Servicios Integrados)
CVM: Customer Value Management (Gestion del Valor de Cliente)

Tabla de ilustraciones

Bibliografía

Chris (buscocrm). *Buscocrm.* s.f. http://www.buscocrm.com/top-10-open-source-crm-systems.php (último acceso: 30 de 06 de 2015).

Conference des Statisticiens Europeens. *Scribd.* 04 de octubre de 2012. http://es.scribd.com/doc/108915633/Defina-que-es-una-Bases-de-Datos-segun-estos-autores-y#scribd (último acceso: 27 de Junio de 2015).

CRMespañol. *CRMespañol.* s.f. http://www.crmespanol.com/links.htm (último acceso: 30 de 06 de 2015).

DuBois, Paul. *MySQL, Fifth Edition.* Addison-Wesley Professional, 2013.

Foster, Elvis C., y Shripad V. Godbole. *Database Systems.* Apress, 2014.

J., Hernandez Michael. *Database Design for Mere Mortals.* Addison-Wesley Professional, 2013.

Kotler, Philip. *Slideshare.* 29 de marzo de 2011. http://es.slideshare.net/MARISOLABIGAIL/seminario-crm-7439611 (último acceso: 26 de junio de 2015).

Lemay, Laura, y Rafe Colburn. *Sams Teach Yourself Web Publishing with HTML and CSS in One Hour a Day: Includes New HTML5 Coverage, Sixth Edition.* Sams, 2010.

Lorna, Jane Mitchell. *PHP Web Services.* O'Reilly Media, Inc, 2013.

Master-Net, Diccionario técnico. *CRM, Cual es la definición?* 21 de 06 de 2010. https://jaimeospina.wordpress.com/category/crm/page/2/ (último acceso: 28 de 06 de 2015).

McFarland , Sawyer David, y Chris Grover. *Dreamweaver CC: The Missing Manual, 2nd Edition.* O'Reilly Media, Inc., 2014.

Morales, Jorge Daniel Anguiano. *IBM.* 30 de 06 de 2014. http://www.ibm.com/developerworks/ssa/data/library/tipos_bases_de_datos/index.html (último acceso: 28 de 06 de 2015).

Moreira Gibaja, Valentin. «Cámara de Valencia - Articulos de Tecnologías de la Información por Latencia SL.» *Las aplicaciones web en el entorno empresarial.* Febrero de 2009. http://coleccion.camaravalencia.com (último acceso: 26 de Junio de 2015).

Olga Pons, Nicolas Marin, Juan Miguel Medina, Silvia Acid, y Mª Amparo Vila. *Introducción a las Bases de Datos El Modelo Relacional.* Madrid: Paraninfo, 2005.

Stephens, Rod. *Beginning Database Design Solutions.* Wrox, 2008.

sugarCRM. s.f. http://www.sugarcrm.com/es/micro (último acceso: 30 de 06 de 2015).

Wenz, Christian. *PHP and MySQL™ Phrasebook.* Addison-Wesley Professional, 2012.